NASA-CR-205796

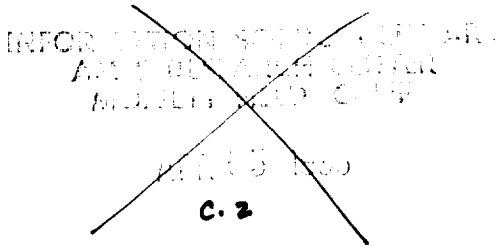
Department of AERONAUTICS and ASTRONAUTICS
STANFORD UNIVERSITY

SUDAAR 631

EXPERIMENTS IN NONLINEAR ADAPTIVE CONTROL OF MULTI-MANIPULATOR, FREE-FLYING SPACE ROBOTS

Vincent Wei-Kang Chen

Department of Electrical Engineering
Stanford University
Stanford, California 94305



Research supported by
AFOSR contract F33615-85-C-5106 and NASA contract NCC 2-333-14

December 1992

EXPERIMENTS IN NONLINEAR ADAPTIVE CONTROL OF MULTI-MANIPULATOR, FREE-FLYING SPACE ROBOTS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Vincent Wei-Kang Chen

December 1992

This dissertation was typeset using the LaTeX document preparation system in conjunction with emTeX running under MS-DOS®. Figures were created with CorelDRAW!™ 3.0 running under Microsoft® Windows™ 3.1 on a 486-based PC. Images were produced using a Hewlett-Packard ScanJet IIc with DeskScan II scanning software also running under Windows. Plots were generated from data collected with Real Time Innovations StethoScope™ and processed using MATLAB™, a product of The MathWorks, Inc. All figures, images, and plots were incorporated as Encapsulated Postscript® files resulting in a completely electronically reproducible document. The final version was produced on a HP LaserJet III printer with the LaserMaster WinJet™800 accelerator board to achieve PostScript outputs at 800dpi resolution.

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Robert H. Cannon, Jr.
Department of Aeronautics and Astronautics
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Gene F. Franklin
Department of Electrical Engineering

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Stephen M. Rock
Department of Aeronautics and Astronautics

Approved for the University Committee on Graduate Studies:

~

Abstract

Sophisticated robots can greatly enhance the role of humans in space by relieving astronauts of low level, tedious assembly and maintenance chores and allowing them to concentrate on higher level tasks. Robots and astronauts can work together efficiently, as a team; but the robot must be capable of accomplishing complex operations and yet be easy to use. Multiple cooperating manipulators are essential to dexterity and can broaden greatly the types of activities the robot can achieve; adding adaptive control can ease greatly robot usage by allowing the robot to change its own controller actions, without human intervention, in response to changes in its environment. Previous work in the Aerospace Robotics Laboratory (ARL) have shown the usefulness of a space robot with cooperating manipulators. The research presented in this dissertation extends that work by adding adaptive control.

To help achieve this high level of robot sophistication, this research made several advances to the field of nonlinear adaptive control of robotic systems. A nonlinear adaptive control algorithm developed originally for control of robots, but requiring joint positions as inputs, was extended here to handle the much more general case of manipulator endpoint-position commands. A new system modelling technique, called *system concatenation* was developed to simplify the generation of a system model for complicated systems, such as a free-flying multiple-manipulator robot system. Finally, the *task*-space concept was introduced wherein the operator's inputs specify only the robot's *task*. The robot's subsequent autonomous performance of each task still involves, of course, endpoint positions and joint configurations as subsets.

The combination of these developments resulted in a new adaptive control framework that is capable of continuously providing full adaptation capability to the complex space-robot system in all modes of operation. The new adaptive control algorithm easily handles free-flying systems with multiple, interacting manipulators, and extends naturally to even larger systems.

The new adaptive controller was experimentally demonstrated on an ideal testbed in the ARL—a first-ever experimental model of a multi-manipulator, free-flying space robot that is capable of capturing and manipulating free-floating objects without requiring human assistance. A graphical user interface enhanced the robot usability: it enabled an operator situated at a remote location to issue high-level task description commands to the robot, and to monitor robot activities as it then carried out each assignment autonomously.

To my grandmother,
whom I have known all my life,
but never really knew.

To my parents,
who have strived always to provide me with the freedom
to choose my own path.

To my wife,
for her boundless understanding and wisdom.

Acknowledgements

I wish to thank my principal advisor, Professor Robert H. Cannon, Jr., for his enthusiastic support and guidance. His energetic efforts in recruiting outstanding students and his exceptional ability for securing funding for the Aerospace Robotics Laboratory (ARL) foster an unsurpassed research environment. Professor Cannon's desire to provide the best tools makes performing research in ARL fun.

To Professor Gene F. Franklin, I wish to extend my gratitude for the stimulating conversations and his insights into adaptive control and beyond. I also thank Professor Stephen M. Rock for his refreshing points of view and encouragement.

This research would not have been possible without the Multiple-Manipulator Free-Flying Space Robot, designed and constructed by Marc Ullman and Ross Koningstein. I am grateful also for the friendship of Marc, not to mention his assistance and counsel. Thanks, too, are due to Stan Schneider for developing the software environment for real-time control; Bill Dickson for willingly assuming the responsibility of the maintenance chores for the robots; Howard Wang for updating the graphical user-interface; Rob Zanutta, Larry Alder, Chris Uhlik, and Dan Rovner for delving into adaptive control discussions; Dennis Morse for his photographic skill and the voluminous pictures of the experimental hardware; Denny, Dave Meer, Steve Ims, and Howard Wang for keeping our computers running; Bill Ballhaus, Edward Wilson, and Richard Marks for providing on-line entertainment.

I also wish to thank Gad Shelef for the mechanical design of the payload, Tom Hasler for his expert workmanship, and Godwin Zhang for the construction of the analog electronics.

Finally, I wish to express my appreciation for the Air Force Office of Scientific Research (AFOSR) and the National Aeronautics and Space Administration (NASA) for providing funding for this research.

$\sim$

Contents

List of Figures

List of Symbols

italic	Italics denote scalar quantities
bold	Boldface denotes vector or matrix quantities
[ ]	Brackets denote mathematical vector or matrix quantities formed by its contents
$\hat{\cdot}$	The hat notation denotes an estimated quantity
$\cdot_d$	A subscripted d denotes a desired quantity
$\tilde{\cdot}$	The tilde notation denotes an error quantity (error = desired - actual)
$'$	The prime notation is used to distinguish between two similar quantities
$\boldsymbol{\Gamma}$	Matrix of adaptive parameter-update gains
$\boldsymbol{\theta}$	Mathematical vector of system parameters
σ_{max}	Maximum singular value
σ_{min}	Minimum singular value
$\boldsymbol{\tau}$	Mathematical vector of actuator torques and forces
$\boldsymbol{\omega}$	Physical angular velocity vector
$\boldsymbol{\omega}_r$	rth partial angular velocity
$\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$	Set of mutually orthogonal unit vectors fixed in inertial space
$\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$	Set of mutually orthogonal unit vectors fixed in robot base
$\mathbf{C}(\mathbf{q}, \mathbf{u})$	Matrix representing Coriolis and centrifugal terms in equations of motion
$\mathbf{f}_{int}$	Mathematical vector of internal forces
$\mathbf{F}$	Mathematical vector of generalized forces

$\mathcal{F}$	Mathematical vector of pseudo-generalized forces
$\mathbf{G}(\mathbf{q})$	Mathematical vector of gravitational terms in equations of motion
$\mathbf{H}^{B/Q}$	Physical angular momentum vector of body B about point Q
$\mathbf{H}_r^{S/S^*}$	rth partial angular momentum vector
$\mathbf{I}_{n\times n}$	$n \times n$ identity matrix
$\mathbf{J}(\mathbf{q})$	Jacobian matrix
$\mathcal{J}(\mathbf{q})$	Generalized Jacobian matrix
$\mathcal{J}^C(\mathbf{q})$	Generalized constraint Jacobian matrix
$\mathbf{K}_P$	Matrix of proportional gains
$\mathbf{K}_V$	Matrix of derivative gains
L	Lagrangian $(T - U)$
$\mathbf{L}$	Physical linear momentum vector
$\mathbf{L}_r$	rth partial linear momentum vector
$\mathbf{M}(\mathbf{q})$	Mass/Inertia matrix in equations of motion
q_r	rth generalized coordinate
$\mathbf{q}$	Mathematical vector of all generalized coordinates
$\mathbf{R}$	Physical force vector
t	Time
T	Kinetic energy
$\mathbf{T}$	Physical torque vector
u_r	rth generalized speed
$\mathbf{u}$	Mathematical vector of all generalized speeds
U	Potential energy
$\mathbf{v}$	Physical velocity vector
$\mathbf{v}_r$	rth partial velocity

$V(t)$	Lyapunov function (scalar)
$\mathbf{W}$	Transformation matrix
$\mathbf{Y}(\mathbf{q}, \mathbf{u}, \mathbf{u}, \dot{\mathbf{u}})$	Motion matrix in the alternate representation for equations of motion. Also known as the regressor.
$\mathbf{x}$	Physical position vector
$\mathbf{y}^{task}$	Mathematical vector of *task*-space control quantities

List of Acronyms

Robotics and Control

ACV	Air-Cushioned Vehicle
DOF	Degrees Of Freedom
EOM	Equations Of Motion
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
INS	Inertial Navigation System
PD	Proportional and Derivative Control
PID	Proportional, Integral, and Derivative Control
PLL	Phase-Locked Loop
PWM	Pulse-Width Modulation
ROV	Remotely-Operated Vehicle
SCARA	Selective Compliance Assembly Robot Arm

NASA Terminology

GAS	Get Away Special
GPS	Global Positioning System
FTS	Flight Telerobotics Servicer
EVA	Extra-Vehicular Activity

MMU	Manned Maneuvering Unit
OMV	Orbital Maneuvering Vehicle
ORU	Orbital Replacement Unit
SRMS	Shuttle Remote Manipulator System

Computers and Electronics

ADC (A/D)	Analog-to-Digital Converter
DAC (D/A)	Digital-to-Analog Converter
CPU	Central Processing Unit
CCD	Charge-Coupled Device
LED	Light Emitting Diode
FIFO	First-In, First-Out
FSM	Finite State Machine
GUI	Graphical User Interface
IC	Integrated Circuit
PCB	Printer-Circuit Board
RAM	Random Access Memory
RVDT	Rotary Variable Differential Transformer
TTL	Transistor-Transistor Logic

Organizations

AIAA	American Institute of Aeronautics and Astronautics
ANSI	American National Standards Institute
ASME	American Society of Mechanical Engineers
ARL	Aerospace Robotics Laboratory
IEEE	Institute of Electrical and Electronic Engineers

ISO	International Standards Organization
NASA	National Aeronautics and Space Administration
SUDAAR	Stanford University Department of Aeronautics and Astronautics Report
WRS	Wind River Systems

Robot Components

GVS	Global Vision System
IOTM	Input/Output Transition Module
LVS	Local Vision System
PCU	Power Control Unit

Chapter 1

Introduction

This dissertation describes theoretical and experimental research on the nonlinear adaptive control of a free-flying space robot with cooperating manipulators. The research was conducted in the Aerospace Robotics Laboratory (ARL) at Stanford University from 1986 to 1992.

1.1 Motivation

Space presents new and exciting challenges. One challenge involves the construction and maintenance of large space structures. While the space program can rely solely on astronauts to perform these duties, doing so makes very inefficient use of their abilities. Robots will add great capabilities to the space program, but only if they possess sufficient dexterity and skill. Multiple cooperative manipulators are essential to dexterity; adaptive control helps significantly to provide the skill.

1.1.1 What is Adaptive Control?

Adaptive control can change its controller actions to assure that the system continues to perform at its best despite changes in the environment or to unknown payload and robot parameters. That is, it *adapts*.

Adaptive control for robots is useful in several important, common situations: 1) When there is poor or no knowledge of the payload parameters, 2) When there are incomplete models of the robot, 3) When there are changes in the environment. While a robust, nonadaptive, controller may provide the same protections as an adaptive controller in these situations, it typically does so with substantially

reduced performance. The added complexity of an adaptive controller wins back that lost performance.

The most obvious situation in which to use adaptive control is for handling payloads that have unknown or poorly known physical properties—for example, when handling damaged satellites where the nature and extent of the damage are unknown. More generally, this capability relieves astronauts of the duty to inform the robot of the detailed physical properties of each payload the robot is to handle. While a comprehensive parts database can relieve much of this responsibility, adaptive control provides protection in cases when the database is not completely accurate or is lacking.

Adaptive control also eases the basic controller design process. There are typically many aspects of the robot itself that are either poorly modelled or not modelled. It is difficult to develop accurate models for robots. In many cases, it is impossible to perform system identification to verify models of space systems on the ground[1] By providing appropriate adjustable parameters to the controller, adaptive control can adapt to model uncertainties to render their effects unimportant.

Another important benefit of adaptive control in space robots is its ability to adapt to the gradual changes that are inevitable in all mechanical systems. Even if friction characteristics are well-characterized at the outset, they will not remain so as the robot ages. A deterministic change—such as inertial property variations as a robot uses up fuel—also can benefit from adaptive control. The controller will adaptively update the controller to track the changes without user intervention or preplanned gain scheduling. By adapting to system changes with time, servicing and reprogramming of the robots can be reduced significantly.

1.1.2 What is *Task*-Space Control?

As a space robot carries out a typical operation, it needs to employ many control modes. For instance, in acquiring a part for assembly, a free-flying robot would start typically in a joint-control mode, where each joint is controlled to a known, "home" location. It then would enter base-control mode as the robot thrusts to approach the part. When the part comes into view of the local cameras or sensors, the robot would utilize manipulator-endpoint-control to track and grasp the part with the arm end effectors. After grasping the part, the robot would switch to cooperative *object*-control mode[2]. After assembling

[1]The Space Shuttle Remote Manipulator System (RMS), for example, cannot support its own weight on the ground.

[2]Cooperative *object*-control should be distinguished from master/slave control, where command inputs must be given to one "master" manipulator, and the other manipulator or manipulators follow the "master" according to some heuristics, typically ignoring the dynamics of the payload. In object-control, the command inputs are specified directly in terms of the desired motions of the *object*. The manipulators then cooperatively effect the requested object motion, taking into account

and releasing the part, the robot might reenter joint-control while returning home or proceeding with the next operation.

Rather than distinguishing the different control modes and supplying different controllers for each control mode, this thesis unifies all the control modes into the *task*-space framework. The *task*-space concept is a generalization of control modes. *Task*-space encompasses all control objectives, such as object position, end-effector position, or joint positions. Choosing a particular *task*-space control vector is tantamount to choosing the control mode. A *task*-space control vector, however, does not have to be restricted to a single type of control mode. A *task*-space control vector for a multiple manipulator robot, for example, can represent endpoint control for one manipulator *and* joint control for the other. As another example, a free-flying space robot can choose a *task*-space control vector to represent the control of the positions of both the payload object and the free-flying base.

This thesis develops a *task*-space controller that effects control for any choice of *task*-space control vector. Because the controller is formulated in *task*-space, it treats all control modes equally. Switching control modes essentially entails switching only the *task*-space control vector. The basic controller structure is unchanged, and a smooth transition is achieved.

1.2 Research Goals

The goal of this research effort is to develop an adaptive control framework that is sophisticated enough to control a free-flying space robot, yet broad enough to be generally applicable. The resulting adaptive control must meet the following requirements:

- The adaptive control must furnish adaptation and control to multiple, cooperating manipulators from a free-flying base and also to simpler, single-manipulator fixed-base robots within the same framework.
- The adaptive control must be able to supply object-level control during cooperative manipulation.
- The adaptive control must provide *total* system adaptation.
- The adaptive control must have *task*-space control capability to provide multiple control modes and to allow "graceful" transitions between control modes.

the dynamics of the complete system.

- The adaptive control must be *implementable.*

Multiple manipulators vastly increase the numbers and types of tasks a robot can perform. They provide dexterity well beyond what a single manipulator is capable of furnishing. Two manipulators grasping a long object at different locations provides better positioning accuracy than a single manipulator grasping one end or the middle of the object. Moreover, smaller motors in cooperating manipulators can provide the same torque capabilities as much larger motors of a single manipulator system.

It is equally important that, in providing multiple-manipulator cooperative control, the adaptive algorithm be able to handle object-level control, wherein the operator directs the motion, or specifies the destination, of the manipulated *object.* This eliminates the need for the physically and mentally demanding tasks of hand-in-glove teleoperation. It permits the human operator to focus totally on the task to be done.

Total system adaptation is an important goal to achieve. It is more than just "payload adaptation" or just "manipulator adaptation". Most existing nonlinear adaptive controllers for robots are geared toward identifying either the robot's physical parameters, or unknown payload parameters, but not both. To provide maximum flexibility, the new adaptive controller needs to have the capability of adapting to both types of parameter changes simultaneously, without resorting to separate adaptation controllers. The adaptive algorithm also should be able to *distinguish* robot parameters from payload parameters, to allow for more intelligent setting of adaptation gains. Since robot parameters typically are better known and change more slowly, heavier weighting placed on adapting to changes in unknown payload parameters permits faster and more accurate adaptation, while still allowing adaptation to changes in the robot.

Adaptive control should provide *task*-space control capability. Doing so makes adaptive control available at all times throughout a complex operation and ensures that the transitions between control modes are smooth.

Because the adaptive algorithm must be implementable, the algorithm must not overwhelm the computational capabilities of present computers. The algorithm must be simple, but effective; and it must extend easily to even larger systems.

1.3 Contributions

In meeting the challenging set of research goals, this research has generated an adaptive control framework that is very general and easily extensible to even larger, more complex systems than the free-flying robot with cooperating manipulators for which it was developed. Yet, this adaptive control framework is equally applicable to simple, single-arm fixed-base robots.

This research takes advantage of the nonlinear, joint-space adaptive controllers already developed for single-arm, fixed-base robots [6]. Because a useful robot is expected to be capable of many control modes—including object-based control, endpoint control and joint control—this research defines *task*-space control—encompassing all control modes—and extends the class of adaptive algorithms from joint-control to *task*-space control, capable of controlling in any mode utilizing the same adaptive control framework. To ease implementation of the new *task*-space adaptive controller for multiple cooperative manipulators, a new modelling technique called *system concatenation* is also developed. The combination of *task*-space adaptive control and *system concatenation* results in a generalized adaptive control framework for rigid-link robotic system.

The research contributions described in this dissertation include:

- *Framework.* Development of a new general adaptive control framework—the adaptive *task*-space framework—that is capable of providing full adaptation to a free-flying space robot with two cooperating manipulators in all modes of operation. The generality allows the adaptive algorithm to extend readily beyond the scope of a single space robot to handle larger systems, including, for example, multiple robots with any number of manipulators.

- *Task Space.* Extension and generalization of a joint-space, nonlinear, adaptive control algorithm, based on inverse dynamics, to control in the *task* space, which represents a broader class of control inputs, including, but not limited to, cooperative *object* control, as well as endpoint control and joint control. This extension provides a means for the adaptive controller to operate in all robot control modes.

- *System Concatenation Method.* Formulation of the *system concatenation* approach for efficient, incremental generation of system models for multiple, interacting systems. *System concatenation* takes full advantage of models already developed for each manipulator or robot subsystem to minimize the additional effort in deriving the total system models used for adaptation. This

formulation eliminates the need to develop closed-kinematic-chain equations of motion for multiple manipulators grasping a common object; and it decreases model complexity and increases computational speed without loss in fidelity.

- *Graphical-User-Interface Integration.* Full integration of the new adaptive control algorithm into a hierarchical control architecture that includes a graphical user interface and a finite-state-table programming environment. The graphical user interface provides the user with an easy method for directly specifying object motions without engaging in fine manipulation details, while the finite-state-table programming provides a degree of autonomy by handling real-time interactions and necessary control-mode transitions to achieve a requested task. Modular design for the adaptive controller allows it to integrate with this hierarchical control architecture without losing its flexibility in handling multiple control modes.

- *Experimental Verification.* Experimental verification of the new adaptive controller, in the hierarchical control environment, on the Multi-Manipulator Free-Flying Space Robot of the the Aerospace Robotics Laboratory (ARL), is shown in Figure 1.1. The robot utilizes an air bearing to reproduce the zero-g, drag-free conditions of space very accurately in two dimensions. This two-armed space robot is totally self-contained, carrying on-board fuel, power, computers, and wireless communications. It is capable of executing semi-autonomous tasks at the direction of a user situated at an off-board computer workstation.

- *Vision System.* The author's development, for the ARL, of the "Point-Grabber II" vision system that, together with software drivers developed in ARL, is capable of tracking bright spots at 60 Hz with better than 1/20 pixel resolution. Duplicates of this high-speed vision system serve as a surrogate global positioning sensor, and as a local on-board end-point sensor[3].

1.4 Review of Related Research

The new *task*-space adaptive control for a space robot presented in this thesis successfully integrates three control disciplines: cooperative manipulation, space robot control, and nonlinear adaptive control. Accordingly, this review of the related research is divided into three, albeit intersecting, sections.

[3]This system has been adopted generically for experiments throughout the laboratory.

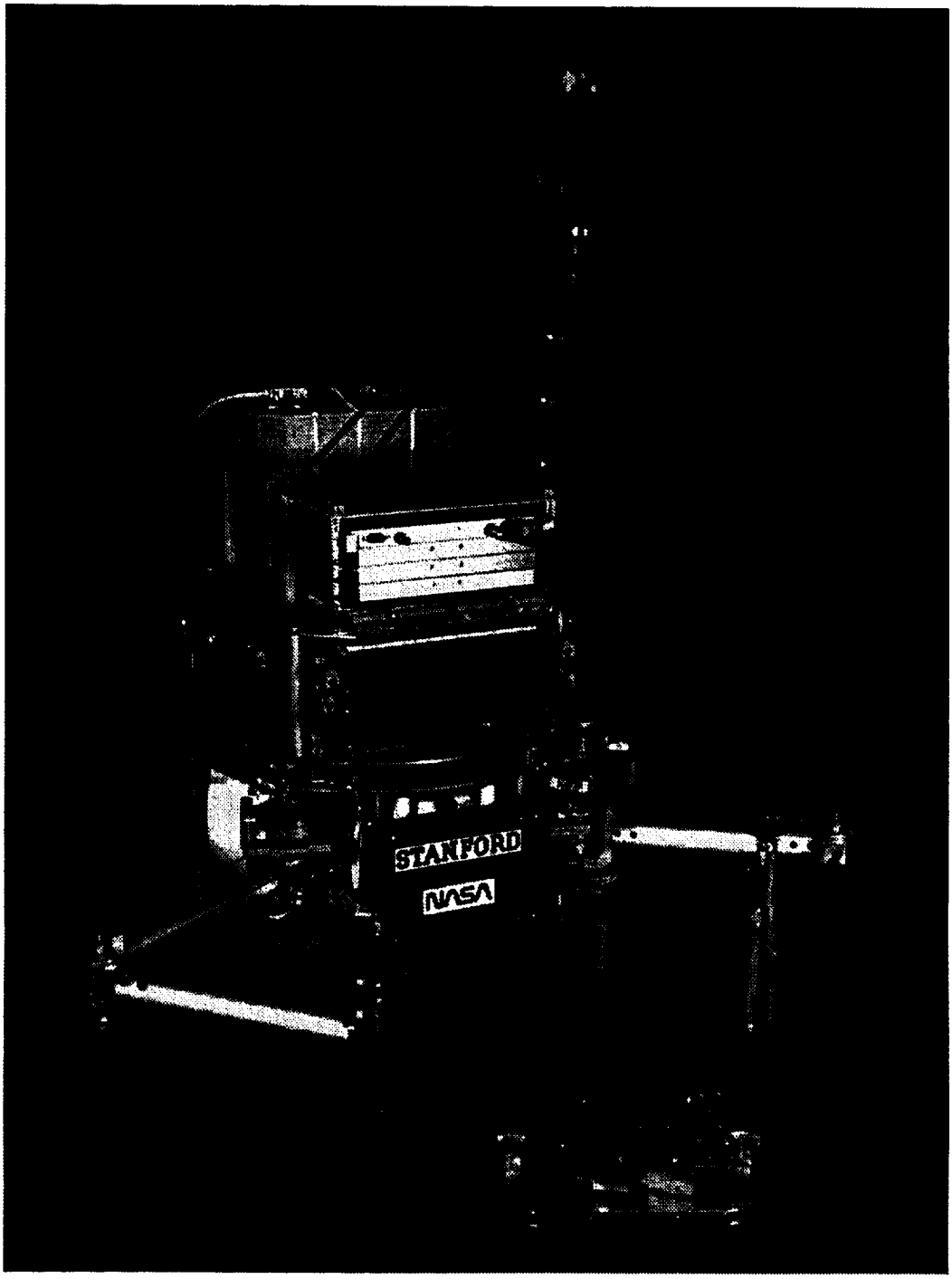


Figure 1.1: **Multi-Manipulator Free-Flying Space Robot**

This experimental space robot is used to verify the performance of the new adaptive controller developed in this thesis. The robot uses an air cushion to faithfully simulate the drag-free conditions of space in a two-dimensional plane.

1.4.1 Object-Based Cooperative Manipulation

The most promising *cooperative* multiple manipulator control strategies involve object motion control. They are promising because of successful experimental demonstrations, and because of the easy interface they provide to higher level controllers; specifically, the higher level controller needs to specify only the object behavior, without having to worry about detailed manipulator motions.

There are roughly three categories of object motion control: object position control, hybrid object force/position control, and object impedance control. With object position control, only the object's position is controlled[25, 21], providing a mechanically "stiff" system. This can be problematic when coming into contact with the environment[4]. Hybrid object force/position control controls the object's position in certain directions, while controlling the object's contact forces in other directions [41, 40, 12, 51, 16]. Hybrid schemes, however, are difficult to implement, requiring control "mode" changes as the object comes into contact, and the continual updating of the force and position degrees of freedom. On the other hand, object impedance control, developed by Schneider at ARL, provides a "compliant" response at all times by maintaining a relationship between the object's position, velocity, and force on the environment [33, 32, 46]. No control switching is necessary as the object comes into contact with its environment.

Schneider [32] provides a detailed review of nonadaptive multiple-manipulator control approaches.

1.4.2 Nonadaptive Space Robotics Research

Alexander [1] pioneered at ARL the research in high-fidelity space robotics experimentation with a demonstration of the control of a single-arm free-floating space robot. He did not utilize thrusters. The controller is essentially a computed-torque controller, with special partitioning of the inertia and nonlinear matrices. This partitioning, unfortunately, is difficult to extend to multiple-arm control. It is even more difficult to extend to adaptive control, because the partitioning must be performed symbolically to determine the adaptable parameters.

Umetani and Yoshida [44, 45] introduced the Generalized Jacobian for maintaining zero momentum of a free-flying base while controlling its single manipulator. Their development did not consider multiple manipulators, and did not extend the Generalized Jacobian beyond zero momentum control.

[4] Rather than a smooth contact, the "stiffness" may cause the object bounce off the contacting surface, possibly damaging the object.

Umetani and Yoshida demonstrated their controller experimentally.

Carignan [8] also performed experiments on manipulation from a free-flying robot using a sliding-mode controller. This controller only partially compensated for the interactions between the manipulator and the robot body. Unfortunately, the performance was limited by the experimental hardware.

Koningstein and Ullman [20, 19] developed at ARL the System Jacobian for controlling a free-flying robot with multiple manipulators. The System Jacobian is essentially a variation of the Umetani and Yoshida's Generalized Jacobian that allows the control of the free-flying robot base and the handling of the constraints in a closed kinematic chain during cooperative manipulation. The System Jacobian also allows the control of the system momentum to be something other than zero. The controller is computed-torque, but implemented via Koningstein's efficient recursive algorithm.

Ullman [43] further extended the work to a full hierarchical controller, incorporating graphical user interface (GUI), state-transition modules. Ullman's research culminated in enabling the space robot to track, chase, and capture free-flying objects and deliver them to operator-specified locations, all with very simple operator commands. Dickson [10] made additional advances by developing a multiple-*robot* controller: Each space robot possesses a pair of arms; and they cooperatively manipulate a single object.

A more detailed review of space robotics research can be found in Ullman [43].

None of the research described in this section incorporates adaptive control. The goal of this dissertation is to develop an adaptive control strategy that will retain *all* the capabilities of Ullman's controller for a free-flying robot with multiple, cooperating manipulators.

1.4.3 Nonlinear Adaptive Robot Control

Craig [9] pioneered the research in nonlinear adaptive control for robotic manipulators. He developed a computed-torque adaptive controller that utilized the standard computed-torque controller as the standard controller block. A drawback of the adaptive portion of the controller is that it required measured joint accelerations and a time-consuming matrix inverse of the system inertia matrix. Craig successfully demonstrated the computed-torque adaptive algorithm on a PUMA arm. The computed-torque adaptive controller applies only to single manipulators under joint control.

Slotine and Li [37, 39] derived the sliding-mode adaptive controller that does not require the

inverse of the inertia matrix. This controller is also an inverse-dynamics algorithm that attempts to cancel directly the nonlinear dynamics of the manipulator. The Lyapunov-based stability proof indicates that the tracking error approaches zero by "sliding" along a surface in a multi-dimensional phase space. Additional research extended the algorithm by adding exponentially forgetting least-squares [22] to improve convergence when there is not enough excitation. Slotine also extended the algorithm to handle endpoint, Cartesian-space control [38]. Slotine and Li have shown impressive capabilities of this adaptive controller experimentally, but, again, only for single fixed-base manipulators under joint control.

Bayard and Wen [48, 6] developed theoretically an entire class of nonlinear adaptive control algorithms, again based on inverse dynamics. They have demonstrated that the new class of algorithms incorporates Craig's computed-torque adaptive algorithm. Bayard and Wen's algorithms also, in general, do not require the inverse of the inertia matrix for adaptation. Although these algorithms are more general, they have been met with much less enthusiasm than Slotine's sliding-mode adaptive control, mainly because of the complexity of the Lyapunov stability proofs. This added complexity does, however, allow them to prove the stability for a broad class of adaptive algorithms. Again, Bayard and Wen developed their algorithms for a single robotic manipulator under joint control. They did not perform any experiments.

Hsia [13] and Ortega and Spong [26] provide good reviews of the field of adaptive control for single, rigid robots. They compare and contrast the various approaches, and provide unified views of these adaptive algorithms.

Zanutta [50] presented the only published experimental research on adaptive control of *multiple cooperating* manipulators, which he did at ARL. He utilized Schneider's object impedance control [32] in conjunction with an inverse-dynamics-based adaptive control that is similar to Slotine's sliding-mode adaptive control. The object impedance control development, however, separates control into two parts: the calculation of necessary forces to effect desired object motion, and the calculation of required actuator torques to effect those forces at the grasp points of the manipulator. The adaptive control applies only to the first stage, adapting to unknown payloads. Adaptation to unknowns in the manipulators must be performed separately, by another means. Zanutta experimentally demonstrated the adaptive object impedance controller on a pair of fully-cooperating manipulators operating from a fixed base.

Additionally, Kim, Walker and Dionise [18] and Hu and Goldenberg [14] presented computer simulations of adaptive control of multiple manipulators. Kim *et. al.* relied on a master/slave configuration for the controller, and involved a complex set of equations and constraints to handle the closed kinematic loops formed by the two manipulators holding an object. Hu and Goldenberg considered a hybrid position- and force-control scheme, but required the inversion of the inertia matrix.

The research presented in this thesis draws on many of the ideas from these researchers. It builds and extends Bayard and Wen's class of adaptive algorithms to handle endpoint control—an extension that is then further developed to handle the more general *task*-space control. Koningstein's method of handling a closed kinematic chain in a control system is generalized, and culminates in the *system concatenation* concept. The Generalized Jacobian ideas of Umetani and Yoshida, and the Jacobian augmentation methods of Koningstein, Ullman, et. al., are further extended to the *task* space.

A final note: In the system modelling arena, parallel theoretical research by Meldrum [23, 24] develops efficient, order-(N) recursive algorithms for computing control and adaptive update equations. These algorithms are based on the spatial operator ideas with Kalman filtering techniques developed by Rodriquez [29, 30].

1.5 Reader's Guide

This thesis is organized into ten chapters and five appendices. Chapter 1 contains the motivation, research goals, contributions, a review of related research, and this Reader's Guide.

Chapter 2 develops the system modelling. It introduces the notation used throughout the dissertation. Chapter 3 describes the control approach by reviewing a nonlinear adaptive control method developed by Bayard and Wen while they were at JPL. This joint-based adaptive controller serves as a starting point for the research presented in this thesis. The chapter discusses the basic elements of this joint-based adaptive algorithm, and highlights the salient aspects of the Lyapunov-based stability and convergence proof.

Chapters 4, 5, and 6 describe developments that generalize the adaptive control framework: extensions of the joint-based adaptive controller, *task*-space concept, and *system concatenation*. Although the latter two new developments can also stand on their own in a nonadaptive environment, their concepts enable the generalization of the adaptive control framework without greatly increasing the complexity

of resulting algorithms.

Chapter 7 merges the developments to formally present the *task*-space adaptive controller, and describes its properties.

Chapter 8 presents the experimental system, and briefly describes the hierarchical control architecture. This chapter outlines the process of integrating the new adaptive controller into the experimental system.

Chapter 9 presents experimental results using the new adaptive algorithm. The controller performance is compared with that of nonadaptive controllers operating either with accurate or with incomplete knowledge of system parameters.

Chapter 10 brings this dissertation to an end with conclusions drawn from the research and suggestions for future research directions.

Appendix A provides more detailed calculations used in the Lyapunov stability proofs. Appendix B includes the user's manual for the Point Grabber II vision system, developed by the author. It describes the design and the software algorithms that enable it to achieve high-resolution sensing. Appendix C details the calibration procedures for calibrating the vision system and motor torque constants, and other sensors. Appendix D supplies the state transition diagrams used for each space robot task. Finally, Appendix E lists the controller input files, representing system parameters, sensor calibration values, controller gains and adaptive gains.

Chapter 2

Modelling

This chapter introduces the notation used in this thesis, and it establishes the mathematical model, i.e., equations of motion, used to describe a robotic system. Section 2.1 presents the model most commonly utilized to describe rigid-link robots. Section 2.2 describes an alternate parameterization of the model that aids adaptive control. This parameterization, shown by An, et. al., [2] to be applicable to rigid robots, isolates the physical parameters of the robotic system into a single mathematical vector[1]. A simple example of a planar two-link manipulator serves to illustrate key concepts throughout the thesis.

2.1 System Model

The equations of motion of a manipulator system consisting of rigid bodies with n degrees of freedom (DOF) can be described by the following form:

$$\mathbf{F} = \mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{C}(\mathbf{q}, \mathbf{u})\mathbf{u} + \mathbf{G}(\mathbf{q}) \tag{2.1}$$

where $\mathbf{q} \in \mathbb{R}^n$ is a mathematical vector of generalized coordinates[2], $\mathbf{u} \in \mathbb{R}^n$ is a mathematical vector of generalized speeds, $\mathbf{F} \in \mathbb{R}^n$ is the mathematical vector of generalized active forces[3], $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$

[1] A mathematical vector is a collection of scalar terms placed into a vector notation. This should be distinguished from a *physical* vector, which describes a direction and magnitude in three-dimensional space. Mathematical vector and matrix notation can greatly simplify the form of the equations presented in this thesis.

[2] Generalized coordinates are a convenient set of coordinates that is sufficient to describe the configuration of a system. Although these convenient coordinates are typically measurable quantities, such as angles and lengths, they do not have to be. In the modelling of systems with distributed flexibility, for example, generalized coordinates "measure" the mode shapes of the system, which are difficult to determine with a ruler and protractor.

[3] This is distinguished from a *physical* force vector. Each element of the generalized force represents the sum of the components of all *physical* forces effecting motion in a particular degree of freedom.

is the symmetric, positive-definite inertia matrix, $\mathbf{C}(\mathbf{q}, \mathbf{u}) \in \mathbb{R}^{n \times n}$ is the matrix of Coriolis and centrifugal terms, and $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^n$ is the vector of gravity torques. The actual terms of the mass matrix, the Coriolis and centrifugal matrix, and the gravity torque vector depend on the choice of the generalized coordinates and generalized speeds. Additionally, for each choice of generalized speeds, the matrix $\mathbf{C}(\mathbf{q}, \mathbf{u})$ is not uniquely defined, although the vector $\mathbf{C}(\mathbf{q}, \mathbf{u})\mathbf{u}$ is unique. An appropriate choice of $\mathbf{C}(\mathbf{q}, \mathbf{u})$ can greatly aid the development of an adaptive controller, as Chapter 3 will describe.

The definition for generalized speeds is [15]:

$$\mathbf{u} \triangleq \mathbf{W}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{W}_t(\mathbf{q}) \tag{2.2}$$

where $\mathbf{W}(\mathbf{q})$ is an $n \times n$ matrix of functions of $\mathbf{q}$ and time t, and $\mathbf{W}_t(\mathbf{q})$ is an $n \times 1$ vector of functions of $\mathbf{q}$ and time t. The $\mathbf{W}_t(\mathbf{q})$ term is nonzero only if there are time-dependent and uncontrolled forcing functions, which are rare in robotics.

In robotics, the common choice for the generalized coordinates, $\mathbf{q}$, corresponds with the joint angles or positions of a manipulator, and $\mathbf{u}$ corresponds with the joint speeds[4], $\dot{\mathbf{q}}$. The generalized forces, $\mathbf{F}$, are assumed to be equal to the actuator torques or forces, $\boldsymbol{\tau}$, arriving at the familiar form:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) \tag{2.3}$$

Strictly speaking, however, $\boldsymbol{\tau}$ is itself a generalized quantity, and depends on the actuator arrangement for a manipulator. Each term of $\boldsymbol{\tau}$ represents only the *net* applied force or torque for each degree of freedom—forces and torques that may be supplied by a combination actuators.

One benefit of the more generalized expression of Equation (2.1) is that appropriate choices for generalized speeds can ease implementation by simplifying the individual terms in the system matrices[5], $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \mathbf{u})$, $\mathbf{G}(\mathbf{q})$. In robots with revolute joints, it is beneficial to choose generalized speeds to correspond with absolute rotational speeds of the links rather than relative speeds of each link with respect to its inboard link. In this special case,

$$\mathbf{u} = \mathbf{W}\dot{\mathbf{q}} \tag{2.4}$$

where $\mathbf{W}$ is a constant matrix of zeros and ones. For a thorough discussion of generalized coordinates and generalized speeds, see [15].

[4] In this situation $\mathbf{W}(\mathbf{q})$ is the (constant) identity matrix, and $\mathbf{W}_t(\mathbf{q})$ is zero.

[5] Essentially, the kinematic configuration of a mechanical system may be developed utilizing a convenient set of coordinates, and the dynamic model may be developed utilizing a convenient set of generalized speeds, where the generalized speeds need not be restricted to be just the time derivatives of the chosen set of coordinates.

Example

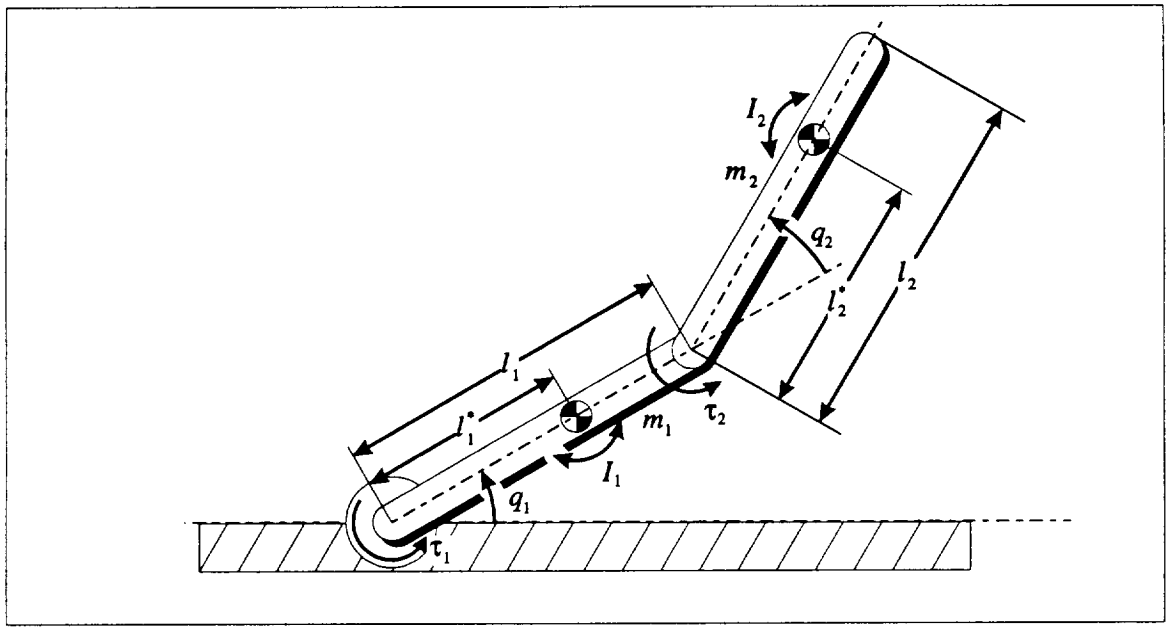


Figure 2.1: **Planar Two-Link Fixed-Base Manipulator**

q_1 and q_2 represent the shoulder and elbow joint angles. m_1, I_1, and l_1 are the mass, moment of inertia, and length of the upper link, while m_2, I_2, and l_2 are those of the lower link. l_1^* and l_2^* represent the center of mass locations of each link, which, for simplicity, are assumed to be along the central axis of each link.

The simple planar two-link manipulator system illustrates the differences between these two forms for the equations of motion. The same example will be used throughout the rest of the thesis for clarifying new concepts. Figure 2.1 introduces the two-link arm system, the notation used for the physical parameters of the system, and the definitions of the generalized coordinates. The more standard equations of motion used by roboticists in for form of Equation (2.3) can be written as:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} m_1 l_1^{*2} + m_2 l_2^{*2} + m_2 l_1^2 \\ +2m_2 l_1 l_2^* \cos(q_2) + I_1 + I_2 \end{pmatrix} & m_2 l_2^{*2} + m_2 l_1 l_2^* \cos(q_2) + I_2 \\ m_2 l_2^{*2} + m_2 l_1 l_2^* \cos(q_2) + I_2 & m_2 l_2^{*2} + I_2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}$$
$$+ \begin{bmatrix} -m_2 l_1 l_2^* \sin(q_2)\dot{q}_2 & -m_2 l_1 l_2^* \sin(q_2)(\dot{q}_1 + \dot{q}_2) \\ m_2 l_1 l_2^* \sin(q_2)\dot{q}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \tag{2.5}$$

or

$$\tau = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$$

Because the arm operates in a plane perpendicular to the direction of gravity, there are no gravitational torques.

Defining the generalized speeds as the absolute angular rotation of each link gives:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_1 + \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \tag{2.6}$$

The generalized equations of motion, in the form of Equation (2.1), simplifies to:

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} m_1 l_1^{*2} + m_2 l_1^2 + I_1 & m_2 l_1 l_2^* \cos(q_2) \\ m_2 l_1 l_2^* \cos(q_2) & m_2 l_2^{*2} + I_2 \end{bmatrix} \begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \end{bmatrix} + \begin{bmatrix} 0 & -m_2 l_1 l_2^* \sin(q_2) u_2 \\ m_2 l_1 l_2^* \sin(q_2) u_1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.7}$$

or

$$\mathbf{F} = \mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{C}(\mathbf{q}, \mathbf{u})\mathbf{u}$$

in which each term of $\mathbf{F}$ represents the total torque applied to each manipulator link, rather than the torque of each actuator:

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} \tau_1 - \tau_2 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{2.8}$$

2.2 Alternate Parameterization

Researchers have shown that the dynamics of serial-kinematic-chained rigid robots allows an alternate parameterization of the equations of motion [2, 17, 4]. This *linear-in-the-parameters* formulation[6] is well suited for adaptive control. It separates all the physical parameters of the system—the masses,

[6] More precisely, this is a parameterization that is linear in terms of a set of physical parameters of the system. The equations of motion are *unchanged* and remain *nonlinear*. The "parameterization" is simply another way of writing the *same* set of equations of motion. This property can be shown relatively easily by examining the Newton-Euler equations for the robotic system (see [2]). It is not clear whether this property is preserved in general for robotic systems that include closed kinematic chains.

moments of inertia, centers of mass locations, link lengths—from the states of the system, resulting in a single vector with p parameters, $\boldsymbol{\theta} \in \mathbb{R}^p$:

$$\mathbf{F} = \mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{C}(\mathbf{q}, \mathbf{u})\mathbf{u} + \mathbf{G}(\mathbf{q}) = \mathbf{Y}(\mathbf{q}, \mathbf{u}, \mathbf{u}, \dot{\mathbf{u}})\boldsymbol{\theta} \tag{2.9}$$

where $\mathbf{Y}(\mathbf{q}, \mathbf{u}, \mathbf{u}, \dot{\mathbf{u}}) \in \mathbb{R}^{n \times p}$ is the *regressor* matrix[7], containing functions of the generalized coordinates, generalized speeds and their time derivatives. An adaptive controller needs to adjust only this single parameter vector to adjust its model of the system.

Example

Utilizing the *linear-in-the-parameters* formulation, the equations of motion[8] for the simple planar arm example, using the more standard notation for robotics given by Equation (2.3), can be expressed as:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \underset{(2.5)}{=} \begin{bmatrix} \ddot{q}_1 & \ddot{q}_1 + \ddot{q}_2 & \begin{pmatrix} 2\cos(q_2)\ddot{q}_1 + \cos(q_2)\ddot{q}_2 \\ -\sin(q_2)\dot{q}_1\,(2\dot{q}_1 + \dot{q}_2) \end{pmatrix} \\ 0 & \ddot{q}_1 + \ddot{q}_2 & \cos(q_2)\ddot{q}_1 + \sin(q_2)\dot{q}_1^2 \end{bmatrix} \begin{bmatrix} m_1 l_1^{*2} + m_2 l_1^2 + I_1 \\ m_2 l_2^{*2} + I_2 \\ m_2 l_1 l_2^* \end{bmatrix}$$
$$= \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\theta} \tag{2.10}$$

or alternatively using generalized speeds and total torques:

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \underset{(2.7)}{=} \begin{bmatrix} \dot{u}_1 & 0 & \cos(q_2)\dot{u}_2 - \sin(q_2)u_2^2 \\ 0 & \dot{u}_2 & \cos(q_2)\dot{u}_1 + \sin(q_2)u_1^2 \end{bmatrix} \begin{bmatrix} m_1 l_1^{*2} + m_2 l_1^2 + I_1 \\ m_2 l_2^{*2} + I_2 \\ m_2 l_1 l_2^* \end{bmatrix}$$
$$= \mathbf{Y}(\mathbf{q}, \mathbf{u}, \mathbf{u}, \dot{\mathbf{u}})\boldsymbol{\theta} \tag{2.11}$$

For both representations,

$$\boldsymbol{\theta} = \begin{bmatrix} m_1 l_1^{*2} + m_2 l_1^2 + I_1 \\ m_2 l_2^{*2} + I_2 \\ m_2 l_1 l_2^* \end{bmatrix} \tag{2.12}$$

is the parameter vector. The number of model parameters in $\boldsymbol{\theta}$, three, does not necessarily match the number of physical parameters, six; and the model parameters can be complicated combinations of

[7]The two u's in the notation for the regressor, $\mathbf{Y}(\mathbf{q}, \mathbf{u}, \mathbf{u}, \dot{\mathbf{u}})$, correspond to the u's in the expression, $\mathbf{C}(\mathbf{q}, \mathbf{u})\mathbf{u}$. It is sometimes useful to distinguish between the two vectors to emphasize that $\mathbf{Y}(\mathbf{q}, \mathbf{u}, \mathbf{u}, \dot{\mathbf{u}})$ is nonlinear in $\mathbf{u}$. Additionally, controllers that choose to use a control such as $\mathbf{F} = \mathbf{M}(\mathbf{q})\dot{\mathbf{u}}_d + \mathbf{C}(\mathbf{q}, \mathbf{u})\mathbf{u}_d + \mathbf{G}(\mathbf{q})$ can be represented by $\mathbf{F} = \mathbf{Y}(\mathbf{q}, \mathbf{u}, \mathbf{u}_d, \dot{\mathbf{u}}_d)\boldsymbol{\theta}$. See [48] for a variety of different of controllers.

[8]Please note that the equations of motion are still *nonlinear*.

the physical parameters. Additionally, utilizing the model parameters, which has rank 3, one cannot uniquely determine all six physical parameters. This example illustrates that it is therefore often not possible to identify the individual link masses or moments of inertias or lengths from the model parameters.

2.3 Summary

This chapter provided two parameterizations for the equations of motion of robotic systems: the standard formulation used by most roboticists and the *linear-in-the-parameters* formulation. The *linear-in-the-parameters* formulation extracts the model parameters into a single mathematical vector and, thus, is well suited for adaptive control. Hence, the adaptive control algorithms presented in this dissertation takes full advantage of this alternate parameterization.

Chapter 3

Control Approach

This chapter lays the groundwork for the development of the new *task*-space adaptive controller presented later in the thesis. It also presents the key components of the stability and convergence proof that will be extended from joint-space control to *task*-space control. Section 3.1 introduces the set of control and adaptation laws, developed by Bayard and Wen, that serves as a basis for this research. The control laws are inverse-dynamics based, similar to the computed-torque control law; the adaptive parameter-update laws are derived from Lyapunov analysis to guarantee stability. Section 3.2 describes the properties of the controller, and Section 3.3 concludes with a stability and convergence proof, based on a Lyapunov function. The limitation to joint-space control of this baseline adaptive control algorithm will be extended in subsequent chapters.

3.1 A Specific Adaptive Control Algorithm

The Bayard and Wen adaptive control algorithms for robotic manipulators represent a class of inverse-dynamics-based controllers [5]. Bayard and Wen have shown, with appropriate choices of the control laws, that this class of algorithms incorporate Craig's computed-torque adaptive controller [9]. Most controllers in their class of algorithms, however, are easier to implement than the computed-torque adaptive controller, because they do not require measured accelerations or the inverse of the inertia matrix. In computer simulations, the Bayard and Wen algorithm also showed less sensitivity to sampling rate and velocity measurement noise than the Slotine and Li sliding-mode adaptive control [7]. Additionally, Bayard and Wen have shown that their algorithms, in the nonadaptive case, are

exponentially stable [48].

This thesis concentrates on extending the Bayard and Wen algorithms to *task*-space control needed by the space robot. It should be noted, however, that similar extensions to the other adaptive algorithms will make them equally suitable for *task*-space control.

Because most of the robotic adaptive control literature uses the more specific form of equations of motion defined by Equation (2.3), initial development of the adaptive control algorithm in this thesis will also use this form. Chapter 4 will generalize the results to utilize generalized speeds for equations of motion given by Equation (2.1).

The development of the adaptive algorithm is illustrated with the following control law[1].

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d + \mathbf{G}(\mathbf{q}) + \mathbf{K}_V\dot{\tilde{\mathbf{q}}} + \mathbf{K}_P\tilde{\mathbf{q}} \tag{3.1}$$

where $(\cdot_d)$ are the desired trajectory quantities and

$$\tilde{(\cdot)} \triangleq (\cdot_d) - (\cdot) \tag{3.2}$$

are the trajectory errors. $\mathbf{K}_P > \mathbf{0}$ is an $n \times n$ diagonal, position gain matrix, and $\mathbf{K}_V > \mathbf{0}$ is an $n \times n$ diagonal, velocity gain matrix. This is an inverse-dynamics controller, where the first three terms of Equation (3.1) represent feedforward[2] terms that cancel much of the dynamics of the plant, and the last two terms effectively provide proportional-derivative (PD) feedback to the system to account for residual trajectory-tracking errors.

For adaptive control when the parameters are unknown, $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}_d)$, and $\mathbf{G}(\mathbf{q})$ are replaced by their estimates, $\widehat{\mathbf{M}}(\mathbf{q})$, $\widehat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}}_d)$, and $\widehat{\mathbf{G}}(\mathbf{q})$. Using these estimated quantities and the *linear-in-the-parameters* parameterization given in Equation (2.9), the adaptive control of Equation (3.1) is modified to:

$$\boldsymbol{\tau} = \widehat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_d + \widehat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d + \widehat{\mathbf{G}}(\mathbf{q}) + \mathbf{K}_V\dot{\tilde{\mathbf{q}}} + \mathbf{K}_P\tilde{\mathbf{q}} \tag{3.3}$$

This control law can also be written in terms of the parameter vector as either of the following:

$$\boldsymbol{\tau} \underset{(2.9)}{=} \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)\widehat{\boldsymbol{\theta}} + \mathbf{K}_V\dot{\tilde{\mathbf{q}}} + \mathbf{K}_P\tilde{\mathbf{q}} \tag{3.4}$$

$$= \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)\boldsymbol{\theta} + \mathbf{K}_V\dot{\tilde{\mathbf{q}}} + \mathbf{K}_P\tilde{\mathbf{q}} - \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)\tilde{\boldsymbol{\theta}} \tag{3.5}$$

[1] This is Control Law 5 in [6] or Control Law 9 in [47].

[2] It can be argued that these are actually feedback terms, since they include measured states, $\mathbf{q}$, but this author adopts a definition that a term is "feedback" only if it includes *errors*, i.e., the *differences* between commanded and actual quantities.

where the number, (2.9), beneath the equality sign is an equation reference, and

$$\tilde{\boldsymbol{\theta}} \triangleq \boldsymbol{\theta} - \hat{\boldsymbol{\theta}} \tag{3.6}$$

is the parameter error vector. The last term in Equation (3.5) represents the difference between the nonadaptive and adaptive control laws. Note that all the unknowns in the system are contained in the parameter estimate vector, $\hat{\boldsymbol{\theta}}$.

The parameter adaptation law is:

$$\dot{\hat{\boldsymbol{\theta}}} = \boldsymbol{\Gamma}\mathbf{Y}^T(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)\left(\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}\right) \tag{3.7}$$

where $\boldsymbol{\Gamma} > \mathbf{0}$ is an $r \times r$ diagonal, positive-definite matrix of parameter update gains, and $c > 0$ is a scalar weighting between the velocity and position errors. This adaptive update law results from Lyapunov analysis, and is derived to guarantee the convergence of trajectory-tracking errors.

3.2 Controller Properties

The block diagram in Figure 3.1 illustrates the structure of the adaptive controller. Following are some properties of this controller:

- *Inverse-dynamics controller.* The full nonlinear dynamics of the manipulator system is included in the "Inverse Dynamics" block. These calculations compensate for much of the nonlinear plant behavior, minimizing trajectory errors. This eases the burden on the PD "Feedback" block, allowing for more aggressive PD gains to handle any residual errors. The parameter vector, $\hat{\boldsymbol{\theta}}$, of the "Inverse Dynamics" block is adaptively updated to change the plant model.

- *Joint-space control.* This standard controller applies for only joint control. The feedback is taken in joint-space. The controller thus operates to reduce errors in joint trajectories. The adaptation law also updates the parameters based on joint-space errors.

- *Tracking-error adaptive algorithm.* The Bayard and Wen adaptive algorithms are tracking-error-based algorithms. The algorithms provide asymptotic convergence of the tracking error to zero, but do not guarantee parameter convergence. Parameter convergence requires sufficient excitation in the trajectories. This limitation poses no serious problems for most robotic activities where minimizing tracking error is the prime objective.

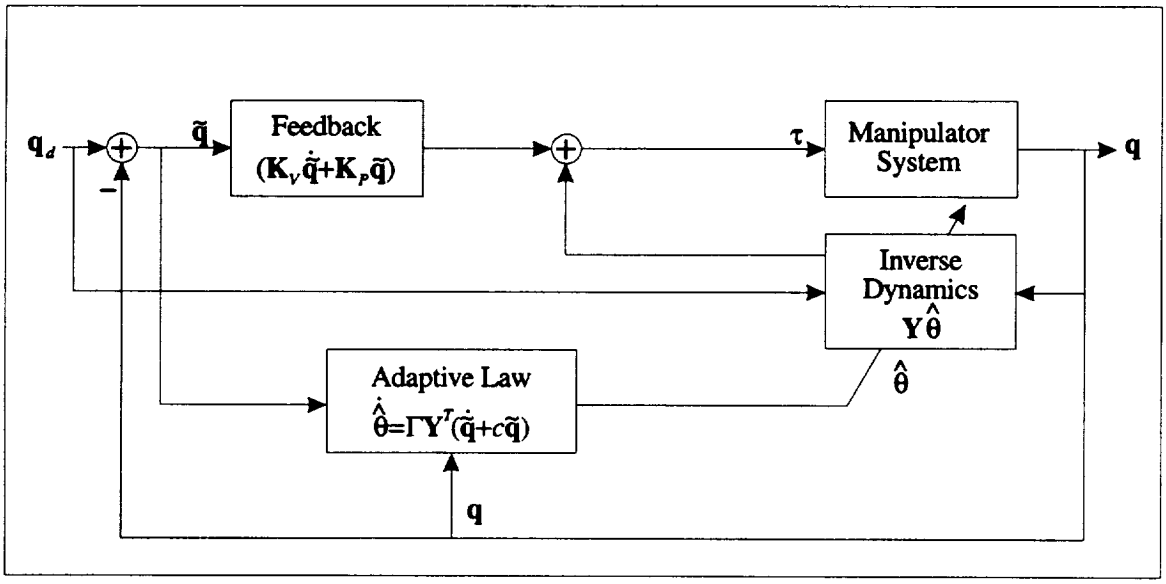


Figure 3.1: **Block Diagram of Bayard and Wen's Adaptive Controller**

This is an inverse-dynamics joint-space controller. The plant model is used to feedforward the nonlinear dynamics to cancel much of the plant nonlinearity. The PD feedback and the adaptive updates are based on the joint trajectory errors.

- *Controller insensitivity to sensor noise.* The inverse dynamics block of the controller uses the *desired* generalized speeds, rather than the measured values. This provides for lower sensitivity to minor sensor noise during regulation. Since there is no requested motion, the desired generalized speeds and accelerations are zero, which in turn causes all terms of the regressor, $\mathbf{Y}(\dot{\mathbf{q}}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)$, to be zero. This effectively disables the inverse dynamics feedforward block. During motion, feedback keeps the actual trajectories close to the desired ones, ensuring that the inverse dynamics block still produces adequate compensation for the nonlinear plant dynamics.

- *Adaptation insensitivity to sensor noise.* Because the parameter adaptation law uses the same regressor, the adaptation is equally insensitive to minor sensor noise during regulation. There is no fear of parameter drift caused by measurement noise or small sensor biases. There is also no need for explicitly enabling and disabling parameter updates during operation.

3.3 Lyapunov Proof

Bayard and Wen proved the stability and convergence of their adaptive algorithms based on a Lyapunov Function [6, 48, 47]. One typically chooses as candidate a Lyapunov Function—a scalar function—to represent the positive-definite "energy" contained in the tracking and parameter estimation errors. Proving that this energy is always decreasing demonstrates that the tracking and estimation errors converge to zero. This is equivalent to showing that the time-derivative of the chosen Lyapunov Function is always negative, which in turn establishes that the system is stable.

Because the development of the *task*-space adaptive controller in subsequent chapters is based on the adaptive algorithm given in this chapter, the stability and convergence proof found in [47] will be repeated here, with some minor corrections and modifications. An outline of the proof describes, in words, the major steps involved in the proof. It also illustrates the typical process of developing Lyapunov proofs. The detailed steps of the stability proof follow in a separate section, with supporting calculations appearing in Appendix A.

3.3.1 Proof Outline

The stability proof starts by examining the controller with exact plant knowledge and no adaptation. Consider the following Lyapunov Function for the system operating with the nonadaptive control law given by Equation (3.1):

$$V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) = \frac{1}{2}\dot{\tilde{\mathbf{q}}}^T \mathbf{M}(\mathbf{q})\dot{\tilde{\mathbf{q}}} + \frac{1}{2}\tilde{\mathbf{q}}^T (\mathbf{K}_P + c\mathbf{K}_V)\,\tilde{\mathbf{q}} + c\tilde{\mathbf{q}}^T \mathbf{M}(\mathbf{q})\dot{\tilde{\mathbf{q}}} \tag{3.8}$$

The first term is square in the joint-velocity tracking error, and as such represents the "energy" in the velocity errors[3]. Similarly, the second term is square in the joint-position tracking errors, and represents the "energy" in the position errors. Since the first two terms are squared terms, they are always positive. The third term is a cross product term between the velocity and position errors—this term is not guaranteed to be positive, so the positive-definiteness of Equation (3.8) must be shown. The scalar, $c > 0$, is a weighting on the cross term—c is the same weighting that appears in the adaptive parameter update law in Equation (3.7).

Noting that $\mathbf{M}(\mathbf{q})$, $\mathbf{K}_P$, and $\mathbf{K}_V$ are symmetric, the time-derivative of the Lyapunov Function can

[3] Recall that $\frac{1}{2}\dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}$ is the *kinetic energy* of the manipulator system, and note the similarity with the first term of the Lyapunov Function of Equation (3.8).

be written as:

$$\begin{aligned}\dot{V}(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t) &= \dot{\tilde{\mathbf{q}}}^T\mathbf{M}(\mathbf{q})\ddot{\tilde{\mathbf{q}}} + \frac{1}{2}\dot{\tilde{\mathbf{q}}}^T\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}} + \dot{\tilde{\mathbf{q}}}^T(\mathbf{K}_P + c\mathbf{K}_V)\tilde{\mathbf{q}} \\ &\quad + c\dot{\tilde{\mathbf{q}}}^T\mathbf{M}(\mathbf{q})\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}^T\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}^T\mathbf{M}(\mathbf{q})\ddot{\tilde{\mathbf{q}}} \end{aligned} \tag{3.9}$$

The stability proof involves showing that the chosen Lyapunov Function, Equation (3.8), is positive-definite; and that its derivative, Equation (3.9), is negative-semi-definite. It follows these steps:

1. Show that the Lyapunov Function, $V(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t)$, is positive definite with the proper choice of c. Determine the appropriate bounds on c: any c that satisfies the bounds can be used in the parameter update law of Equation (3.7).

2. Show that the Lyapunov Function is bounded if the velocity and position errors are bounded.

3. Compute the time-derivative of $V(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t)$ in order to show with the following steps that it is never positive:

   (a) Use the definitions for trajectory errors, equations of motion, and the control law in Equations (3.2, 2.3, 3.4) to expand $\dot{V}(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t)$.

   (b) Define a specific representation of $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ that will eliminate and simplify terms in $\dot{V}(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t)$.

   (c) Determine the bounds on the remaining terms and show $\dot{V}(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t)$ is negative-semi-definite. Doing so places requirements on the gain matrices, $\mathbf{K}_P$ and $\mathbf{K}_V$.

   This proves that if the initial errors are bounded, $V(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t)$ remains uniformly bounded, which in turn implies that the trajectory errors remain uniformly bounded.

4. Given the above, use Barbalat's theorem [27] and the boundedness of $\ddot{\tilde{\mathbf{q}}}$ to show that $\tilde{\mathbf{q}}$ and $\dot{\tilde{\mathbf{q}}}$ tend to zero as $t \rightarrow \infty$, proving stability.

The stability and convergence proof for the *adaptive* controller requires a modification of the Lyapunov Function to include a term that represents the "energy" in the parameter errors:

$$V_1(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},\tilde{\boldsymbol{\theta}},t) = V(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t) + \frac{1}{2}\tilde{\boldsymbol{\theta}}^T\boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\theta}} \tag{3.10}$$

Using the adaptive control law in Equation (3.5) and taking the time-derivative of the new Lyapunov function, $V_1(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, \tilde{\boldsymbol{\theta}}, t)$, yields:

$$\dot{V}_1(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, \tilde{\boldsymbol{\theta}}, t) = \dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) + \tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma}^{-1} \dot{\tilde{\boldsymbol{\theta}}} + \tilde{\boldsymbol{\theta}}^T \mathbf{Y}^T(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d) \left(\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}\right) \tag{3.11}$$

Noting that $\dot{\tilde{\boldsymbol{\theta}}} = -\dot{\hat{\boldsymbol{\theta}}}$ and $\frac{d}{dt}\boldsymbol{\theta} = \mathbf{0}$, substitute the adaptive update law, Equation (3.7), to eliminate the last two terms of Equation (3.11). This reduces the adaptive case to the nonadaptive case of Equation (3.9). Making the same arguments shows that once again, $\tilde{\mathbf{q}}$ and $\dot{\tilde{\mathbf{q}}}$ tend to zero as $t \to \infty$, proving stability for the adaptive algorithm.

3.3.2 Proof Details

This section presents the proof in some detail. The steps follow those derived in [47], but with the notation utilized in this thesis. They also correct some minor errors. It is useful to examine first the structure present in the equations of motion for a system of rigid bodies.

The Lagrangian formulation of the equations of motion starts with the kinetic and potential energies for a system of rigid bodies:

$$\begin{aligned} T &= \tfrac{1}{2}\dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q})\dot{\mathbf{q}} \\ U &= -\mathbf{q}^T \boldsymbol{\tau} + g(\mathbf{q}) \end{aligned} \tag{3.12}$$

where T is the kinetic energy, U is the total potential energy, of which $g(\mathbf{q})$ is the gravitational potential energy component. Using the Lagrangian, $L = T - U$, and applying Lagrange's Equation:

$$\begin{aligned} 0 &= \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial L}{\partial \mathbf{q}} \\ &= \frac{d\left(\mathbf{M}(\mathbf{q})\dot{\mathbf{q}}\right)}{dt} - \frac{1}{2}\frac{\partial\left(\dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q})\right)}{\partial \mathbf{q}}\dot{\mathbf{q}} - \boldsymbol{\tau} + \frac{\partial g(\mathbf{q})}{\partial \mathbf{q}} \\ &= -\boldsymbol{\tau} + \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \frac{1}{2}\left(\frac{\partial\left(\mathbf{M}(\mathbf{q})\dot{\mathbf{q}}\right)}{\partial \mathbf{q}}\right)^T \dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) \end{aligned} \tag{3.13}$$

This results in the equations of motion in the form of Equation (2.3), which can be rewritten as:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} = \boldsymbol{\tau} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q}) \tag{3.14}$$

where

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \triangleq \left(\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) - \frac{1}{2}\left(\frac{\partial\left(\mathbf{M}(\mathbf{q})\dot{\mathbf{q}}\right)}{\partial \mathbf{q}}\right)^T\right)\dot{\mathbf{q}} \tag{3.15}$$

$$\mathbf{G}(\mathbf{g}) \triangleq \frac{\partial g(\mathbf{q})}{\partial \mathbf{q}} \tag{3.16}$$

The stability proof uses the structure of the equations of motion demonstrated by Equations (3.13–3.16).

Step 1: Positive-Definiteness of V.

Define the following notation for bounds on matrices[4]:

$$\begin{aligned}
\mu_M &\triangleq \inf_{\mathbf{q}} \sigma_{min}\left(\mathbf{M}(\mathbf{q})\right) \\
\mu_p &\triangleq \sigma_{min}\left(\mathbf{K}_P\right) \\
\mu_v &\triangleq \sigma_{min}\left(\mathbf{K}_V\right) \\
\gamma_M &\triangleq \sup_{\mathbf{q}} \sigma_{max}\left(\mathbf{M}(\mathbf{q})\right) \\
\gamma_p &\triangleq \sigma_{max}\left(\mathbf{K}_P\right) \\
\gamma_v &\triangleq \sigma_{max}\left(\mathbf{K}_V\right)
\end{aligned}$$

where σ_{min} and σ_{max} denote the minimum and maximum singular values, respectively. Roughly, these singular values give the minimum and maximum "gain" of a matrix[5].

With these definitions, a lower bound on the Lyapunov Function of Equation (3.8) is[6]:

$$V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) \geq \frac{1}{2}\left(\mu_p + c\mu_v\right)\|\tilde{\mathbf{q}}\|^2 + \frac{1}{2}\mu_M \left\|\dot{\tilde{\mathbf{q}}}\right\|^2 - c\gamma_M \|\tilde{\mathbf{q}}\| \left\|\dot{\tilde{\mathbf{q}}}\right\| \tag{3.17}$$

Using "perfect squares", the cross term can be written as:

$$\begin{aligned}
-2c\gamma_M \|\tilde{\mathbf{q}}\| \left\|\dot{\tilde{\mathbf{q}}}\right\| &= c\gamma_M \left(\epsilon \|\tilde{\mathbf{q}}\| - \epsilon^{-1}\left\|\dot{\tilde{\mathbf{q}}}\right\|\right)^2 - c\gamma_M \epsilon^2 \|\tilde{\mathbf{q}}\|^2 - c\gamma_M \epsilon^{-2}\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \\
&\geq -c\gamma_M \epsilon^2 \|\tilde{\mathbf{q}}\|^2 - c\gamma_M \epsilon^{-2}\left\|\dot{\tilde{\mathbf{q}}}\right\|^2
\end{aligned} \tag{3.18}$$

[4]The inf notation represents *infimum*, which is the lower bound of the specified set, but this bound does not have to be a member of the set. For example, $\inf 1/x = 0$, but 0 does not belong to the set $\{1/x\}$. Similarly, sup represents the *supremum*, or the upper bound of the specified set.

[5]The maximum and minimum singular values of a matrix, K, give the maximum and minimum "gain" of K, respectively, in a 2-norm sense. That is,

$$\begin{aligned}
\sigma_{max} &= \sup_{\mathbf{u}} \frac{\|\mathbf{K}\mathbf{u}\|}{\|\mathbf{u}\|} \\
\sigma_{min} &= \inf_{\mathbf{u}} \frac{\|\mathbf{K}\mathbf{u}\|}{\|\mathbf{u}\|}
\end{aligned}$$

[6]The notation, $\|\cdot\|$, denotes the 2-norm, or magnitude, of $(\cdot)$. It is defined by:

$$\|\mathbf{u}\| \triangleq \left(\mathbf{u}^T\mathbf{u}\right)^{\frac{1}{2}}.$$

for any $\epsilon \in \mathbb{R}$. Thus,

$$\begin{aligned} V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) &\underset{(3.17,3.18)}{\geq} \frac{1}{2}\left(\mu_p + c\mu_v - c\gamma_M\epsilon^2\right)\|\tilde{\mathbf{q}}\|^2 + \frac{1}{2}\left(\mu_M - c\gamma_M\epsilon^{-2}\right)\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \\ &\geq \alpha_1\|\tilde{\mathbf{q}}\|^2 + \alpha_2\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \end{aligned} \tag{3.19}$$

where,

$$\begin{aligned} \alpha_1 &= \mu_p + c\mu_v - c\gamma_M\epsilon^2 \\ \alpha_2 &= \mu_M - c\gamma_M\epsilon^{-2} \end{aligned} \tag{3.20}$$

To ensure $V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ is positive definite, α_1 and α_2 must be greater than zero. Solving each inequality in terms of ϵ^2 gives:

$$\begin{aligned} \frac{c\gamma_M}{\mu_M} &< \epsilon^2 < \frac{\mu_p + c\mu_v}{c\gamma_M} \\ \Rightarrow \quad c^2\gamma_M^2 &< \mu_M\left(\mu_p + c\mu_v\right) \end{aligned} \tag{3.21}$$

Solving the quadratic in Equation (3.21) for c yields:

$$c < \frac{\mu_M\mu_v}{2\gamma_M^2}\left(1 + \left(1 + \frac{4\mu_p\gamma_M^2}{\mu_M\mu_v^2}\right)^{\frac{1}{2}}\right) \tag{3.22}$$

Thus, choosing c to satisfy Equation (3.22) guarantees that $V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ is a positive-definite function.

Step 2: Boundedness for V.

Following similar procedures as the above, an upper bound on the Lyapunov Function is:

$$V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) \leq \frac{1}{2}\left(\gamma_p + c\gamma_v\right)\|\tilde{\mathbf{q}}\|^2 + \frac{1}{2}\gamma_M\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 + c\gamma_M\|\tilde{\mathbf{q}}\|\left\|\dot{\tilde{\mathbf{q}}}\right\| \tag{3.23}$$

Using "perfect squares" again, the cross term can be written as:

$$\begin{aligned} 2c\gamma_M\|\tilde{\mathbf{q}}\|\left\|\dot{\tilde{\mathbf{q}}}\right\| &= -c\gamma_M\left(\eta\|\tilde{\mathbf{q}}\| - \eta^{-1}\left\|\dot{\tilde{\mathbf{q}}}\right\|\right)^2 + c\gamma_M\eta^2\|\tilde{\mathbf{q}}\|^2 + c\gamma_M\eta^{-2}\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \\ &\leq c\gamma_M\eta^2\|\tilde{\mathbf{q}}\|^2 + c\gamma_M\eta^{-2}\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \end{aligned} \tag{3.24}$$

for any $\eta \in \mathbb{R}$. Thus,

$$\begin{aligned} V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) &\underset{(3.23,3.24)}{\leq} \frac{1}{2}\left(\gamma_p + c\gamma_v + c\gamma_M\eta^2\right)\|\tilde{\mathbf{q}}\|^2 + \frac{1}{2}\left(\gamma_M + c\gamma_M\eta^{-2}\right)\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \\ &\leq \beta_1\|\tilde{\mathbf{q}}\|^2 + \beta_2\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \end{aligned} \tag{3.25}$$

where,

$$\begin{aligned} \beta_1 &= \gamma_p + c\gamma_v + c\gamma_M\eta^2 \\ \beta_2 &= \gamma_M + c\gamma_M\eta^{-2} \end{aligned} \tag{3.26}$$

Noting that β_1 and β_2 are guaranteed to be greater than zero, Equation (3.25) shows that $V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ is uniformly bounded if the position and velocity errors, $\tilde{\mathbf{q}}$ and $\dot{\tilde{\mathbf{q}}}$, are uniformly bounded.

Assuming that $\|\tilde{\mathbf{q}}(t)\|$ and $\left\|\dot{\tilde{\mathbf{q}}}(t)\right\|$ are uniformly bounded, the lower bound on $V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$, leads to the following:

$$\begin{aligned} V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) &\underset{(3.19)}{\geq} \alpha_1 \|\tilde{\mathbf{q}}(t)\|^2 \\ \Rightarrow \left(\frac{\overline{V}}{\alpha_1}\right)^{\frac{1}{2}} &\geq \sup_t \|\tilde{\mathbf{q}}(t)\| \end{aligned} \tag{3.27}$$

where

$$\overline{V} = \sup_t V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t). \tag{3.28}$$

Step 3: Negative-Semi-Definiteness of $\dot{V}$.

Using a particular representation for the Coriolis and centrifugal matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and the identities in Equations (3.31–3.34), this section proves that $\dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ is never positive, showing that $V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ never increases. Thus if the initial trajectory errors are bounded, they remain uniformly bounded.

Step 3a: Expanding $\dot{V}$. Using the definition for trajectory error in Equation (3.2) to expand $\ddot{\tilde{\mathbf{q}}}$, and equations of motion in Equation (3.14) to substitute for $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}$, and the control law in Equation (3.1) to substitute for τ, the time-derivative of the Lyapunov Function in Equation (3.9) expands to:

$$\begin{aligned} \dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) \underset{(3.2,3.14)}{=} & \dot{\tilde{\mathbf{q}}}^T \left(\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_d - \tau + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \frac{1}{2}\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}} + \mathbf{K}_P\tilde{\mathbf{q}}\right) \\ & + c\dot{\tilde{\mathbf{q}}}^T\mathbf{M}(\mathbf{q})\dot{\tilde{\mathbf{q}}} \\ & + c\tilde{\mathbf{q}}^T\left(\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}} + \mathbf{K}_V\dot{\tilde{\mathbf{q}}} + \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_d - \tau + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})\right) \\ \underset{(3.1)}{=} & \dot{\tilde{\mathbf{q}}}^T \left(-\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d - \mathbf{K}_V\dot{\tilde{\mathbf{q}}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}}\right) \\ & + c\dot{\tilde{\mathbf{q}}}^T\mathbf{M}(\mathbf{q})\dot{\tilde{\mathbf{q}}} \\ & + c\tilde{\mathbf{q}}^T\left(\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d - \mathbf{K}_P\tilde{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}\right) \\ = & -c\tilde{\mathbf{q}}^T\mathbf{K}_P\tilde{\mathbf{q}} - \dot{\tilde{\mathbf{q}}}^T\left(\mathbf{K}_V - c\mathbf{M}(\mathbf{q})\right)\dot{\tilde{\mathbf{q}}} \end{aligned} \tag{3.29}$$

$$+\dot{\tilde{\mathbf{q}}}^T\left(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d+\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}+\frac{1}{2}\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}}\right)$$
$$+c\tilde{\mathbf{q}}^T\left(\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}}-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d+\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}\right) \tag{3.30}$$

Choosing a "good" representation of $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ will help further simplify this expression.

Step 3b: A Representation of C. Begin by first defining a matrix, $\mathbf{M}_D$, such that

$$\mathbf{M}_D(\mathbf{q},\mathbf{y}) \triangleq \sum_{i=1}^{n}\frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\mathbf{y}\mathbf{e}_i^T \tag{3.31}$$

where $\mathbf{e}_i \in \mathbb{R}^n$ is the ith unit vector. Following are some useful identities for $\mathbf{M}_D(\mathbf{q},\mathbf{y})$[7]:

Identity 1:

$$\frac{\partial\left(\mathbf{M}(\mathbf{q})\mathbf{y}\right)}{\partial \mathbf{q}} = \mathbf{M}_D(\mathbf{q},\mathbf{y}) \tag{3.32}$$

Identity 2:

$$\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\mathbf{y} = \mathbf{M}_D(\mathbf{q},\mathbf{y})\dot{\mathbf{q}} \tag{3.33}$$

Identity 3:

$$\mathbf{M}_D(\mathbf{q},\mathbf{y})\mathbf{z} = \mathbf{M}_D(\mathbf{q},\mathbf{z})\mathbf{y} \tag{3.34}$$

Now choose $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ to be the following[8]:

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}}) - \frac{1}{2}\mathbf{M}_D^T(\mathbf{q},\dot{\mathbf{q}}) \tag{3.35}$$

Appendix A.2 shows that this is a valid choice by satisfying the condition in Equation (3.15). Using this representation in the Lyapunov derivative of Equation (3.30) yields:

$$\begin{aligned}
\dot{V}(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t) &\underset{(3.33,3.35)}{=} -c\tilde{\mathbf{q}}^T\mathbf{K}_P\tilde{\mathbf{q}} - \dot{\tilde{\mathbf{q}}}^T\left(\mathbf{K}_V - c\mathbf{M}(\mathbf{q})\right)\dot{\tilde{\mathbf{q}}} \\
&\quad - \left(\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}\right)^T\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d \\
&\quad + \dot{\tilde{\mathbf{q}}}^T\left(\frac{1}{2}\mathbf{M}_D(\mathbf{q},\dot{\tilde{\mathbf{q}}})\dot{\mathbf{q}} + \mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \frac{1}{2}\mathbf{M}_D^T(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}\right) \\
&\quad + c\tilde{\mathbf{q}}^T\left(\mathbf{M}_D(\mathbf{q},\dot{\tilde{\mathbf{q}}})\dot{\mathbf{q}} + \mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \frac{1}{2}\mathbf{M}_D^T(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}\right) \\
&\underset{(3.2,3.34,3.35)}{=} -c\tilde{\mathbf{q}}^T\mathbf{K}_P\tilde{\mathbf{q}} - \dot{\tilde{\mathbf{q}}}^T\left(\mathbf{K}_V - c\mathbf{M}(\mathbf{q})\right)\dot{\tilde{\mathbf{q}}} - \frac{1}{2}\dot{\tilde{\mathbf{q}}}^T\mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\tilde{\mathbf{q}}} \\
&\quad + c\tilde{\mathbf{q}}^T\left(\left(\mathbf{M}_D^T(\mathbf{q},\dot{\mathbf{q}}_d) - \mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}}_d)\right)\dot{\tilde{\mathbf{q}}} - \frac{1}{2}\mathbf{M}_D^T(\mathbf{q},\dot{\tilde{\mathbf{q}}})\dot{\tilde{\mathbf{q}}}\right)
\end{aligned} \tag{3.36}$$

[7] See Appendix A.1 for the proofs.

[8] This representation does not satisfy the skew-symmetric property required by the sliding-mode adaptive controller [37]. See Appendix A.2 for more details.

Step 3c: Upper Bound for $\dot{V}$. This step shows that the upper bound for $\dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ is never positive for proper choices of $\mathbf{K}_P$ and $\mathbf{K}_V$. First determine the upper bound on the last two terms of Equation (3.36):

$$\begin{aligned}\left|-\frac{1}{2}\dot{\tilde{\mathbf{q}}}^T\mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\tilde{\mathbf{q}}}\right| &\underset{(3.31)}{=} \left|\frac{1}{2}\dot{\tilde{\mathbf{q}}}^T\left(\sum_{i=1}^{n}\frac{\partial\mathbf{M}(\mathbf{q})}{\partial q_i}\dot{\mathbf{q}}_d\mathbf{e}_i^T\right)\dot{\tilde{\mathbf{q}}}\right| \\ &\leq \frac{1}{2}\gamma_1\gamma_3\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \end{aligned} \tag{3.37}$$

where

$$\gamma_1 \triangleq \sup_{\mathbf{q}}\left(\sum_{i=1}^{n}\left\|\frac{\partial\mathbf{M}(\mathbf{q})}{\partial q_i}\right\|_2\right) \tag{3.38}$$

$$\gamma_3 \triangleq \sup_{t}|\dot{\mathbf{q}}_d| \tag{3.39}$$

Similarly,

$$\begin{aligned}&\left|c\tilde{\mathbf{q}}^T\left(\left(\mathbf{M}_D^T(\mathbf{q},\dot{\mathbf{q}}_d)-\mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}}_d)\right)\dot{\tilde{\mathbf{q}}}-\frac{1}{2}\mathbf{M}_D^T(\mathbf{q},\dot{\tilde{\mathbf{q}}})\dot{\tilde{\mathbf{q}}}\right)\right| \\ &\leq 2c\gamma_1\gamma_3\|\tilde{\mathbf{q}}\|\left\|\dot{\tilde{\mathbf{q}}}\right\|+\frac{1}{2}c\gamma_1\|\tilde{\mathbf{q}}\|\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \\ &\underset{(3.18)}{\leq} c\gamma_1\gamma_3\xi^2\|\tilde{\mathbf{q}}\|^2+c\gamma_1\gamma_3\xi^{-2}\left\|\dot{\tilde{\mathbf{q}}}\right\|^2+\frac{1}{2}c\gamma_1\|\tilde{\mathbf{q}}\|\left\|\dot{\tilde{\mathbf{q}}}\right\|^2\end{aligned} \tag{3.40}$$

Note that the "perfect squares" expression similar to Equation (3.18) is used once again to eliminate the cross term, $2c\gamma_1\gamma_3\|\tilde{\mathbf{q}}\|\left\|\dot{\tilde{\mathbf{q}}}\right\|$. Combining these results gives:

$$\dot{V}(\tilde{\mathbf{q}},\dot{\tilde{\mathbf{q}}},t) \underset{(3.36,3.37,\,3.40)}{\leq} -\lambda_1\|\tilde{\mathbf{q}}\|^2-\lambda_2\left\|\dot{\tilde{\mathbf{q}}}\right\|^2+\frac{1}{2}c\gamma_1\|\tilde{\mathbf{q}}\|\left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \tag{3.41}$$

where

$$\lambda_1 \triangleq c\left(\mu_p-\gamma_1\gamma_3\xi^2\right) \tag{3.42}$$

$$\lambda_2 \triangleq \left(\mu_v-c\gamma_M-\gamma_1\gamma_3\left(\frac{1}{2}+c\xi^{-2}\right)\right) \tag{3.43}$$

Both λ_1 and λ_2 must be positive. This leads to the condition:

$$\begin{aligned}\frac{1}{2}\frac{c\gamma_1\gamma_3}{\mu_v-c\gamma_M-\frac{1}{2}\gamma_1\gamma_2} &< \xi^2 < \frac{\mu_p}{\gamma_1\gamma_3} \\ \Rightarrow \quad \frac{1}{2}c\gamma_1^2\gamma_3^2 &< \mu_p\left(\mu_v-c\gamma_M-\frac{1}{2}\gamma_1\gamma_2\right) \\ \Rightarrow \quad c &< \frac{\mu_p\left(\mu_v-\frac{1}{2}\gamma_1\gamma_3\right)}{\mu_p\gamma_M+\frac{1}{2}\gamma_1^2\gamma_3^2}\end{aligned} \tag{3.44}$$

Since $c > 0$, the last inequality shows $\mathbf{K}_V$ must be chosen large enough to satisfy:

$$\mu_v > \frac{1}{2}\gamma_1\gamma_3 \tag{3.45}$$

to ensure that $\dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ in Equation (3.41) has some negative definite terms.

Now assume that $\|\tilde{\mathbf{q}}\|$ and $\left\|\dot{\tilde{\mathbf{q}}}\right\|$ are bounded such that $\overline{V}$ defined by Equation (3.28) exists, choose $\mathbf{K}_V$ large enough such that:

$$\lambda_2 - \frac{1}{2}c\gamma_1\left(\frac{\overline{V}}{\alpha_1}\right)^{\frac{1}{2}} > 0 \tag{3.46}$$

Then

$$\overline{\lambda}_2 \triangleq \lambda_2 - \frac{1}{2}c\gamma_1 \|\tilde{\mathbf{q}}\| \underset{(3.27,3.46)}{>} 0 \tag{3.47}$$

The upper bound on $\dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ can now be written as:

$$\dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) \underset{(3.41,3.47)}{\leq} -\lambda_1 \|\tilde{\mathbf{q}}\|^2 - \overline{\lambda}_2 \left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \leq 0 \tag{3.48}$$

Thus given $\mathbf{K}_V$ large enough, $\dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ is never positive, so $V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ is *never* increasing, such that

$$\overline{V} = V(\tilde{\mathbf{q}}(t_0), \dot{\tilde{\mathbf{q}}}(t_0), t_0). \tag{3.49}$$

This in turn implies, from Equation (3.19), that the trajectory errors, $\tilde{\mathbf{q}}$ and $\dot{\tilde{\mathbf{q}}}$, are uniformly bounded.

If the initial errors, $\tilde{\mathbf{q}}(t_0)$ and $\dot{\tilde{\mathbf{q}}}(t_0)$, are small enough—so $\overline{V}$ is small enough—then the assumption in Equation (3.46) can be satisfied easily.

Step 4: Stability

Application of Barbalat's theorem at this point shows that if the trajectory errors remain bounded, then they, in fact, also tend to zero. The theorem is stated as follows [6]:

Barbalat's Theorem

If $W(t)$ is a real function of the real variable t, defined and uniformly continuous for $t \geq 0$, and if the limit of the integral

$$\lim_{t\to\infty} \int_0^t W(t')\, dt'$$

exists and is a finite number, then

$$\lim_{t\to\infty} W(t) = 0.$$

In other words, if the integral of a continuous function approaches a limit and does not grow without bound, then W *must* approach zero, because it cannot continue to add to the integral.

To show stability of the controller, let

$$W(t) \triangleq \lambda_1 \|\tilde{\mathbf{q}}\|^2 + \overline{\lambda}_2 \left\|\dot{\tilde{\mathbf{q}}}\right\|^2 \tag{3.50}$$

such that

$$\dot{V}(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t) \underset{(3.48)}{\leq} -W(t). \tag{3.51}$$

Integrating both sides of Equation (3.51) from 0 to t, yields upon rearranging,

$$\int_0^t W(t')\,dt' \leq V(\tilde{\mathbf{q}}(0), \dot{\tilde{\mathbf{q}}}(0), 0) - V(\tilde{\mathbf{q}}(t), \dot{\tilde{\mathbf{q}}}(t), t) \tag{3.52}$$

Since $V(\tilde{\mathbf{q}}(0), \dot{\tilde{\mathbf{q}}}(0), 0)$ is bounded, and $V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}, t)$ is nonincreasing and bounded below, it follows that

$$\lim_{t\to\infty} \int_0^t W(t')\,dt' < \infty \tag{3.53}$$

Also, since $\ddot{\mathbf{q}}_d$ is chosen to be bounded, and since rigid-body dynamics dictate that $\ddot{\mathbf{q}}$ is bounded for bounded input forces and torques, $\dot{W}(t)$ is bounded, which implies that $W(t)$ is uniformly continuous. Applying Barbalat's Theorem, therefore, gives

$$\begin{aligned} &\lim_{t\to\infty} W(t) = 0 \\ \Rightarrow \quad &\|\tilde{\mathbf{q}}\| \rightarrow 0 \\ &\left\|\dot{\tilde{\mathbf{q}}}\right\| \rightarrow 0 \end{aligned} \tag{3.54}$$

This proves the asymptotic stability of the nonadaptive controller.

3.3.3 Adaptive Controller Stability

As the previous section showed, modifying the Lyapunov function to include the parameter-estimate errors, and applying the adaptive update law of Equation (3.7), makes the above proof equally applicable. Thus this section has also proved the asymptotic stability of the *adaptive* controller.

Finally, it should be noted that the above proof used rather loose bounds to derive requirements on the gains, c, $\mathbf{K}_P$, $\mathbf{K}_V$. These requirements are sufficient, but not necessary, so one can use them for initial design. In practice, the gains that achieve the best controller and adaptation performance may not meet these criteria.

Chapter 4

Adaptive Control Extensions

The Bayard and Wen class of adaptive control algorithms is applicable in a relatively restricted realm—single-arm joint control. This chapter describes the first steps in extending the adaptive control algorithm to *task*-space control: 1) extension to utilize equations of motion developed using generalized speeds to ease the design procedure and 2) extension to endpoint control. These extensions provide the essential building blocks for developing the *task*-space adaptive controller in Chapter 5.

Section 4.1 extends, in a limited fashion, the adaptive algorithm to handle the more generalized modelling method using generalized speeds. This extension benefits controller implementation by simplifying the terms that appear in the equations of motion. The approach is straightforward and, except for a transformation from actuator torques to generalized forces, involves little change from the original adaptive controller.

Section 4.2 further extends the algorithm to handle endpoint control. This extension utilizes the end-effector Jacobian matrix for transforming endpoint motions in Cartesian space to variations in the generalized coordinates and generalized speeds. Once again, the changes to the original controller structure are minor; this section furnishes a summary of the differences.

Section 4.2.3 offers an outline and the detailed stability proof for the new endpoint adaptive controller. The Lyapunov function must be altered for endpoint control, and includes the Jacobian matrix. The proof follows the same steps as the original proof by Wen, with an additional stability requirement that the manipulators must stay away from kinematic singularity. Since endpoint control *per se* also makes this requirement, this condition does not pose a serious limitation to the new endpoint adaptive control algorithm.

4.1 Extension to Generalized Speeds

Appropriate choices for generalized speeds can simplify the terms in the equations of motion for a system. This aids implementation by reducing the number of calculations[1]. For instance, the two-link arm example in Chapter 2 illustrated that for a planar robotic system with revolute joints, it is beneficial to choose generalized speeds that correspond with the absolute rotation rates of the links, rather than the joint rates. This section, therefore, extends the adaptive controller given by Equation (3.4) and Equation (3.7) to this useful, albeit limited, case.

For generalized speeds chosen to correspond with absolute angular rates for the rotational degrees of freedom, the generalized speeds and the joint rates are related by a constant matrix of zeros and ones[2]:

$$\mathbf{u} \triangleq \mathbf{W}\dot{\mathbf{q}} \tag{4.1}$$

where $\mathbf{W}$ is the $n \times n$ constant matrix. The new joint-space adaptive control law can be expressed as:

$$\mathbf{F} = \widehat{\mathbf{M}}'(\mathbf{q})\dot{\mathbf{u}}_d + \widehat{\mathbf{C}}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d + \widehat{\mathbf{G}}'(\mathbf{q}) + \mathbf{K}_V\tilde{\mathbf{u}} + \mathbf{W}^{-T}\mathbf{K}_P\tilde{\mathbf{q}} \tag{4.2}$$

$$\tau = \mathbf{W}^T\mathbf{F} \tag{4.3}$$

or in terms of the parameter vector:

$$\mathbf{F} \underset{(2.9)}{=} \mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\widehat{\boldsymbol{\theta}} + \mathbf{K}_V\tilde{\mathbf{u}} + \mathbf{W}^{-T}\mathbf{K}_P\tilde{\mathbf{q}} \tag{4.4}$$

$$= \mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\boldsymbol{\theta} + \mathbf{K}_V\tilde{\mathbf{u}} + \mathbf{W}^{-T}\mathbf{K}_P\tilde{\mathbf{q}} - \mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\tilde{\boldsymbol{\theta}} \tag{4.5}$$

The corresponding new parameter update law is:

$$\dot{\hat{\boldsymbol{\theta}}} = \Gamma\mathbf{Y}'^T(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\left(\tilde{\mathbf{u}} + c\mathbf{W}^T\tilde{\mathbf{q}}\right) \tag{4.6}$$

where the $'$ notation is used to distinguish the system equations from those derived *not* using generalized speeds. That is, $\mathbf{M}'(\mathbf{q})$, $\mathbf{C}'(\mathbf{q},\mathbf{u})$, $\mathbf{G}'(\mathbf{q})$, and $\mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)$ are the matrices defined by the generalized equations of motion,

$$\mathbf{F} = \mathbf{M}'(\mathbf{q})\dot{\mathbf{u}} + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u} + \mathbf{G}'(\mathbf{q}) = \mathbf{Y}'(\mathbf{q},\mathbf{u},\mathbf{u},\dot{\mathbf{u}})\boldsymbol{\theta}. \tag{4.7}$$

Figure 4.1 shows the block diagram of the new controller. Application of the control law,

[1] The example in Chapter 2 demonstrates this.

[2] See the example given by Equation (2.6).

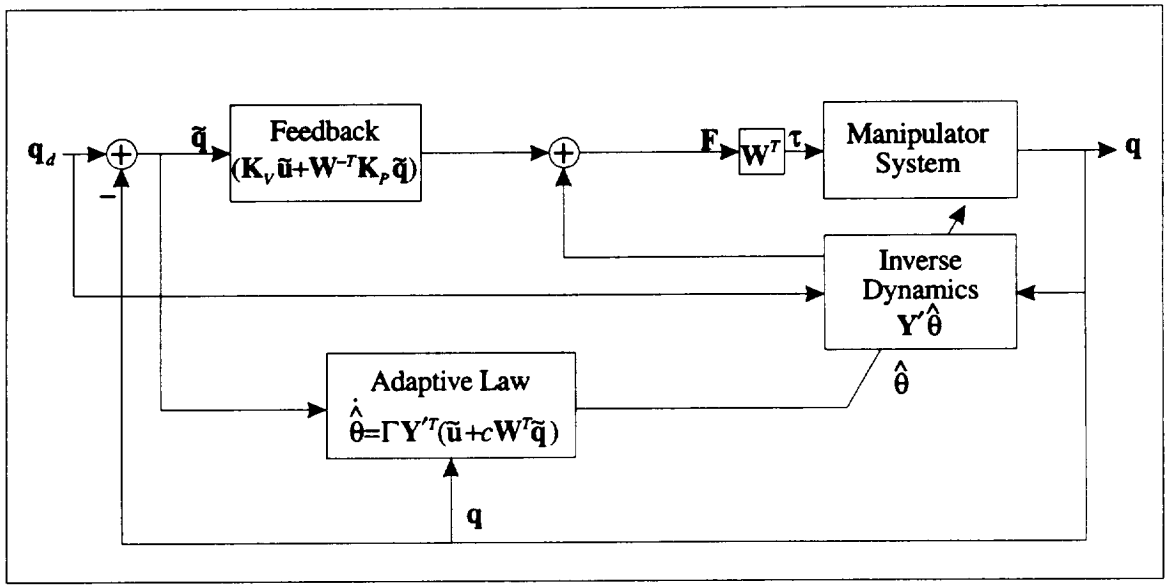


Figure 4.1: **Block Diagram of the Adaptive Controller using Generalized Speeds**

This controller allows the use of a system model that was developed with generalized speeds. The controller structure is almost identical to the original controller, Figure 3.1, with the exception of the transformation block, W, to bring generalized forces into actual joint torques.

Equation (4.4), results in a mathematical vector representing the desired *generalized* forces to apply to the system. Implementation, therefore, requires the transformation of this vector back into actuator torques. Other than this transformation, there is little change from the original adaptive controller.

4.1.1 Transforming Equations of Motion

Examining the relationship between the two forms of equations of motion defined by Equation (2.1) and Equation (2.3) is useful for simplifying the stability proof.

Differentiating the generalized speeds in Equation (4.1) with respect to time yields:

$$\dot{\mathbf{u}} = W\ddot{\mathbf{q}} \tag{4.8}$$

Substituting Equation (4.1) and Equation (4.8) into the equations of motion using $\dot{\mathbf{q}}$s results in:

$$\begin{aligned} \tau &\underset{(2.3)}{=} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) \\ &\underset{(4.1,4.8)}{=} \mathbf{M}(\mathbf{q})\mathbf{W}^{-1}\dot{\mathbf{u}} + \mathbf{C}(\mathbf{q},\mathbf{W}^{-1}\mathbf{u})\mathbf{W}^{-1}\mathbf{u} + \mathbf{G}(\mathbf{q}) \end{aligned} \tag{4.9}$$

Premultiplying both sides of Equation (4.9) by $\mathbf{W}^{-T}$ gives:

$$\mathbf{W}^{-T}\boldsymbol{\tau} = \mathbf{W}^{-T}\mathbf{M}(\mathbf{q})\mathbf{W}^{-1}\dot{\mathbf{u}} + \mathbf{W}^{-T}\mathbf{C}(\mathbf{q}, \mathbf{W}^{-1}\mathbf{u})\mathbf{W}^{-1}\mathbf{u} + \mathbf{W}^{-T}\mathbf{G}(\mathbf{q}) \tag{4.10}$$

Matching terms of the generalized equations of motion in Equation (2.1) with those in Equation (4.10) yields:

$$\begin{aligned} \mathbf{F} &= \mathbf{W}^{-T}\boldsymbol{\tau} \\ \mathbf{M}'(\mathbf{q}) &= \mathbf{W}^{-T}\mathbf{M}(\mathbf{q})\mathbf{W}^{-1} \\ \mathbf{C}'(\mathbf{q}, \mathbf{u}) &= \mathbf{W}^{-T}\mathbf{C}(\mathbf{q}, \mathbf{W}^{-1}\mathbf{u})\mathbf{W}^{-1} \\ \mathbf{G}'(\mathbf{q}) &= \mathbf{W}^{-T}\mathbf{G}(\mathbf{q}) \end{aligned} \tag{4.11}$$

It also follows from Equation (4.10) that,

$$\mathbf{Y}'(\mathbf{q}, \mathbf{u}, \mathbf{u}, \dot{\mathbf{u}})\boldsymbol{\theta} = \mathbf{W}^{-T}\mathbf{Y}(\mathbf{q}, \mathbf{W}^{-1}\mathbf{u}, \mathbf{W}^{-1}\mathbf{u}, \mathbf{W}^{-1}\dot{\mathbf{u}})\boldsymbol{\theta} \tag{4.12}$$

The equations in (4.11) and (4.12) represent the transformation between the two forms of equations of motion and the transformation between generalized forces to actuator torques.

4.1.2 Stability Proof

The stability proof for this case is rather straightforward. Substituting the transformations in Equation (4.11) into the control and adaptive laws of Equation (4.4) and Equation (4.6), results in *exactly* the original Bayard and Wen control and adaptation laws[3]. The *same* Lyapunov Function, Equation (3.10), therefore, can be used to prove stability.

For completeness sake, the Lyapunov Function and its derivative can be written in the $\prime$ notation and in terms of generalized speeds as:

$$V(\tilde{\mathbf{q}}, \tilde{\mathbf{u}}, t) = \frac{1}{2}\tilde{\mathbf{u}}^T\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} + \frac{1}{2}\tilde{\mathbf{q}}^T(\mathbf{K}_P + c\mathbf{K}_V)\tilde{\mathbf{q}} + c\tilde{\mathbf{q}}^T\mathbf{W}^T\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} + \frac{1}{2}\tilde{\boldsymbol{\theta}}^T\boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\theta}} \tag{4.13}$$

and

$$\begin{aligned} \dot{V}(\tilde{\mathbf{q}}, \tilde{\mathbf{u}}, t) &= \tilde{\mathbf{u}}^T\mathbf{M}'(\mathbf{q})\dot{\tilde{\mathbf{u}}} + \frac{1}{2}\tilde{\mathbf{u}}^T\dot{\mathbf{M}}'(\mathbf{q}, \mathbf{u})\tilde{\mathbf{u}} + \dot{\tilde{\mathbf{q}}}^T(\mathbf{K}_P + c\mathbf{K}_V)\tilde{\mathbf{q}} \\ &\quad + c\mathbf{u}^T\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} + c\tilde{\mathbf{q}}^T\mathbf{W}^T\dot{\mathbf{M}}'(\mathbf{q}, \mathbf{u})\tilde{\mathbf{u}} + c\tilde{\mathbf{q}}^T\mathbf{W}^T\mathbf{M}'(\mathbf{q})\dot{\tilde{\mathbf{u}}} \\ &\quad + \tilde{\boldsymbol{\theta}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\theta}}} + \tilde{\boldsymbol{\theta}}^T\mathbf{Y}'^T(\mathbf{q}, \mathbf{u}_d, \mathbf{u}_d, \dot{\mathbf{u}}_d)(\tilde{\mathbf{u}} + c\mathbf{W}\tilde{\mathbf{q}}) \end{aligned} \tag{4.14}$$

Using the transformation equations also will reduce these to the original Lyapunov function and its derivative.

[3] See Appendix A.3 for the proof.

4.2 Extension to Endpoint Control

While endpoint control can benefit a fixed-base industrial robot performing precise, dextrous manipulation, it is crucial for a mobile robot that needs to perform *acquisition* before any manipulation. With no fixed base from which to reference the location of the parts it is to acquire, the mobile robot controller must use feedback directly on the manipulator endpoint positions relative to the part. The space robot capturing a free-flying satellite, for example, must use endpoint control to track the grasp points on the satellite before capture.

Hence, this section further extends the adaptive control algorithm to endpoint control. The desired trajectories a manipulator must follow are given in terms of endpoint Cartesian coordinates. The adaptive law also updates the parameters based on the endpoint tracking errors. Although at this point this extension is still applicable only to a single-manipulator robot, the results can be quite useful for later extension to the full cooperating-manipulator space robot.

4.2.1 Endpoint Jacobian

The extension from joint control to endpoint control utilizes the Jacobian matrix. Figure 4.2 illustrates the relationship between the endpoint and joint positions. In general, the Jacobian matrix arises when differentiating mathematical vector fields[4]. It is composed of the partial derivatives of the vector function. In robotics, the term, *Jacobian*, has taken on a more specific meaning, namely, the relationship between joint rates and the manipulator end-effector velocity (a *physical* vector) and orientation rates.

Simple kinematic analysis gives the endpoint position and orientation as functions of the joint angles, represented by:

$$\mathbf{x} = \mathbf{k}(\mathbf{q}) \tag{4.15}$$

where $\mathbf{x} \in \mathbb{R}^m$ contains the endpoint position and orientation, and $\mathbf{k} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are the kinematic equations. When $m = n$, the robot is said to be nonredundant, and when $m < n$, the robot is redundant.

Taking the time-derivative of both sides of Equation (4.15) gives:

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{k}(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} + \frac{\partial \mathbf{k}(\mathbf{q})}{\partial t} \tag{4.16}$$

[4]Not to be confused with the *physical* position ($\mathbf{x}$) and velocity ($\dot{\mathbf{x}}$) vectors which are of course also the centerpiece of Figure 4.2.

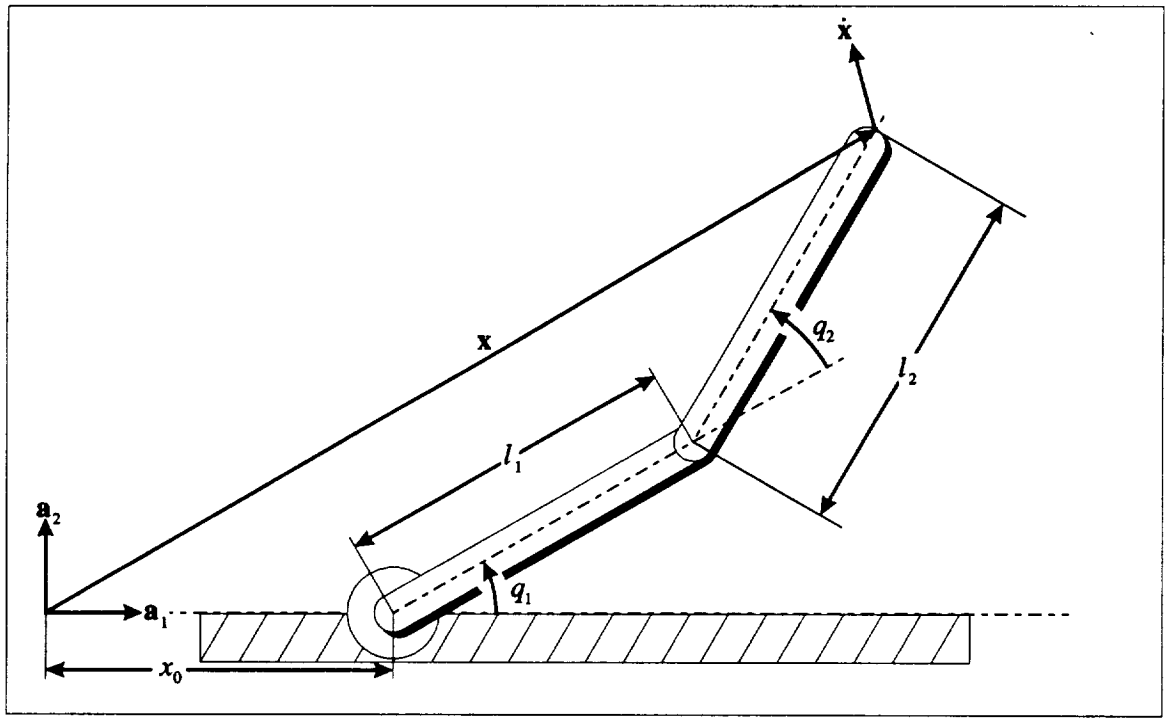


Figure 4.2: **Cartesian Endpoint Position and Velocity**

q_1 and q_2 represent the shoulder and elbow joint angles. $\mathbf{x}$ is the position vector of the manipulator endpoint and $\dot{\mathbf{x}}$ represents the endpoint velocity vector.

Defining the *Jacobian* to be:

$$\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{k}(\mathbf{q})}{\partial \mathbf{q}} \tag{4.17}$$

and a time-dependent term to be:

$$\mathbf{J}_t(\mathbf{q}) = \frac{\partial \mathbf{k}(\mathbf{q})}{\partial t} \tag{4.18}$$

yields the endpoint Jacobian relationship for robotics:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{J}_t(\mathbf{q}) \tag{4.19}$$

When there are no prescribed[5] motions in the system, $\mathbf{J}_t(\mathbf{q})$ is zero, which yields the more common relationship:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{4.20}$$

[5]A prescribed motion is a motion that the system is forced to undergo in response to unmodelled external forcing functions. In robotic systems, where all forces are modelled, there are no prescribed motions.

The inverse Jacobian relationships are:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{x}} - \mathbf{J}^{-1}(\mathbf{q})\mathbf{J}_t(\mathbf{q}) \tag{4.21}$$

$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\left(\ddot{\mathbf{x}} - \dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{x}} - \dot{\mathbf{J}}_t(\mathbf{q})\right) \tag{4.22}$$

To make use of generalized speeds, additionally define a *transformed* Jacobian, $\mathbf{J}'(\mathbf{q})$, as:

$$\mathbf{J}'(\mathbf{q}) \triangleq \mathbf{J}(\mathbf{q})\mathbf{W}^{-1} \tag{4.23}$$

such that the following are true:

$$\mathbf{u} = \mathbf{W}\dot{\mathbf{q}} \tag{4.24}$$

$$\dot{\mathbf{x}} = \mathbf{J}'(\mathbf{q})\mathbf{u} + \mathbf{J}'_t(\mathbf{q}) \tag{4.25}$$

$$\mathbf{u} = \mathbf{J}'^{-1}(\mathbf{q})\dot{\mathbf{x}} - \mathbf{J}'^{-1}(\mathbf{q})\mathbf{J}'_t(\mathbf{q}) \tag{4.26}$$

$$\dot{\mathbf{u}} = \mathbf{J}'^{-1}(\mathbf{q})\left(\ddot{\mathbf{x}} - \dot{\mathbf{J}}'(\mathbf{q},\mathbf{u})\mathbf{J}'^{-1}(\mathbf{q})\dot{\mathbf{x}} - \dot{\mathbf{J}}'_t(\mathbf{q})\right) \tag{4.27}$$

Example

The two-link arm example illustrates that using generalized speeds can also simplify the terms in the Jacobian. The physical position vector for the endpoint can be written as (See Figure 4.2):

$$\mathbf{x} = \left(l_1\cos(q_1) + l_2\cos(q_1+q_2) + x_0\right)\mathbf{a}_1 + \left(l_1\sin(q_1) + l_2\sin(q_1+q_2)\right)\mathbf{a}_2 \tag{4.28}$$

where $\mathbf{a}_1$ and $\mathbf{a}_2$ are mutually orthogonal unit vectors, and x_0 is a constant offset of the shoulder from the origin of the coordinate system. Taking the partial derivative with respect to $\mathbf{q}$ gives the Jacobian:

$$\mathbf{J}(\mathbf{q}) \underset{(4.17)}{=} \begin{bmatrix} -l_1\sin(q_1) - l_2\sin(q_1+q_2) & -l_2\sin(q_1+q_2) \\ l_1\cos(q_1) + l_2\cos(q_1+q_2) & l_2\cos(q_1+q_2) \end{bmatrix} \tag{4.29}$$

where it has been expressed in the coordinate system aligned with $\mathbf{a}_1$ and $\mathbf{a}_2$. The *transformed* Jacobian for use with generalized speeds, utilizing $\mathbf{W}$ from Equation (2.6), is:

$$\mathbf{J}'(\mathbf{q}) \underset{(4.23,2.6)}{=} \begin{bmatrix} -l_1\sin(q_1) & -l_2\sin(q_1+q_2) \\ l_1\cos(q_1) & l_2\cos(q_1+q_2) \end{bmatrix} \tag{4.30}$$

Because there are no prescribed motions, $\mathbf{J}_t(\mathbf{q})$ and $\mathbf{J}'_t(\mathbf{q})$ are zero.

4.2.2 Endpoint Adaptive Control

Using the *transformed* Jacobian matrix defined above, the endpoint adaptive control law can be written in terms of generalized speeds as:

$$\mathbf{F} = \widehat{\mathbf{M}}'(\mathbf{q})\dot{\mathbf{u}}_d + \widehat{\mathbf{C}}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d + \widehat{\mathbf{G}}'(\mathbf{q}) + \mathbf{J}'^T(\mathbf{q})\left(\mathbf{K}_{Vx}\dot{\tilde{\mathbf{x}}} + \mathbf{K}_{Px}\tilde{\mathbf{x}}\right) \tag{4.31}$$

$$\tau = \mathbf{W}^T\mathbf{F} \tag{4.32}$$

or in terms of the parameter vector:

$$\mathbf{F} \underset{(2.9)}{=} \mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\widehat{\boldsymbol{\theta}} + \mathbf{J}'^T(\mathbf{q})\left(\mathbf{K}_{Vx}\dot{\tilde{\mathbf{x}}} + \mathbf{K}_{Px}\tilde{\mathbf{x}}\right) \tag{4.33}$$

$$= \mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\boldsymbol{\theta} + \mathbf{J}'^T\left(\mathbf{K}_{Vx}\dot{\tilde{\mathbf{x}}} + \mathbf{K}_{Px}\tilde{\mathbf{x}}\right) - \mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\tilde{\boldsymbol{\theta}} \tag{4.34}$$

The corresponding adaptive update law is:

$$\dot{\widehat{\boldsymbol{\theta}}} = \boldsymbol{\Gamma}\mathbf{Y}'^T(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\mathbf{J}'^{-1}(\mathbf{q})\left(\dot{\tilde{\mathbf{x}}} + c\tilde{\mathbf{x}}\right) \tag{4.35}$$

In addition, the desired generalized speed and its derivative are given by:

$$\mathbf{u}_d \underset{(4.26)}{=} \mathbf{J}'^{-1}(\mathbf{q})\mathbf{x}_d - \mathbf{J}'^{-1}(\mathbf{q})\mathbf{J}'_t(\mathbf{q}) \tag{4.36}$$

$$\dot{\mathbf{u}}_d \underset{(4.27)}{=} \mathbf{J}'^{-1}(\mathbf{q})\Big(\ddot{\mathbf{x}}_d - \dot{\mathbf{J}}'(\mathbf{q},\mathbf{u})\mathbf{J}'^{-1}(\mathbf{q})\dot{\mathbf{x}} - \dot{\mathbf{J}}'(\mathbf{q},\mathbf{u})\mathbf{J}'^{-1}(\mathbf{q})\mathbf{J}'_t(\mathbf{q}) - \dot{\mathbf{J}}'_t(\mathbf{q})\Big) \tag{4.37}$$

The key differences between the new endpoint adaptive controller and the joint controller can be described as:

- *Endpoint-trajectory tracking.* The controller tracks endpoint trajectories, because it performs feedback directly on the endpoint tracking error. The adaptive parameter update is also based on the endpoint tracking error.

- *Jacobian calculations.* The Jacobian calculation blocks are the only real structural changes in the controller. They are needed to convert the desired endpoint trajectories into equivalent desired joint-space trajectories for use by the inverse-dynamics feedforward block. The Jacobian also translates the feedback forces, expressed in Cartesian space, into the mathematical space represented by the generalized forces. Finally, the adaptive update block also uses the Jacobian to map the tracking errors to the joint space, in which the adaptive model is derived.

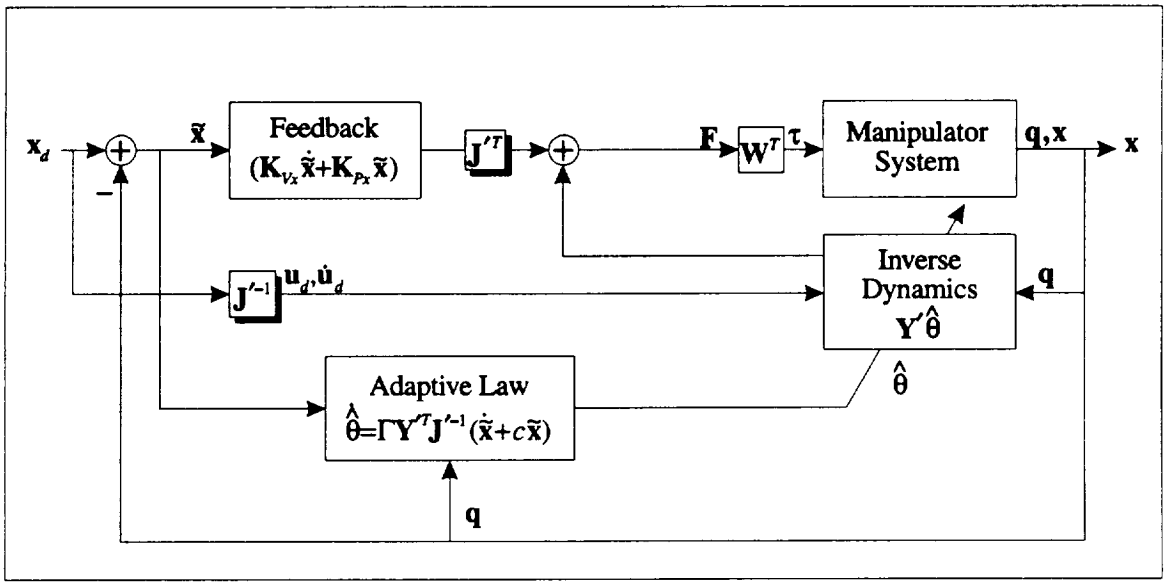


Figure 4.3: **Block Diagram of the Endpoint Adaptive Controller**

The endpoint adaptive control tracks endpoint trajectories by performing feedback directly on the endpoint errors, $\tilde{x}$. Note that the adaptive parameter update is also based on the endpoint tracking error. This controller structure is once again very similar to the original controller, Figure 3.1. The only additions are the Jacobian blocks that transform the endpoint quantities into equivalent joint-level quantities. The transformation block, W, is still required to bring generalized forces into actual joint torques.

- *Torque transformation.* Since the control law is given in terms of generalized speeds, the transformation block, W, is still required to transform the commanded force into actuator torques.

4.2.3 Stability Proof

The stability proof involves choosing a new Lyapunov Function to represent the "energy" in terms of the *endpoint* tracking errors and then showing that it goes to zero, implying that the tracking errors also go to zero. In doing so, the proof places an additional requirement on the controller: the manipulator must remain away from geometric singularity, which equates to the matrix inverse of the Jacobian being bounded. This is a very natural requirement, since the controller becomes ill-behaved near kinematic singularities; it is also a requirement *whenever* one implements a Cartesian-space controller. In practice, kinematic singularity is easy to detect[6]. If it cannot be avoided, the controller can change control modes

[6]Typically, kinematic singularities occur when two or more degrees of freedom of the manipulator "line up", where the manipulator loses degrees of freedom. For the two-link manipulator, kinematic singularity occurs with the arm straight out

to bring the manipulator into a more favorable configuration.

This proof starts with the nonadaptive controller by choosing the Lyapunov Function and calculating its derivative. It then follows steps similar to that of the original proof in Chapter 3:

1. Show that the Lyapunov Function, $V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$, is positive definite with the proper choice of c. Determine the appropriate bounds on c.

2. Show that the Lyapunov Function is bounded if the velocity and position errors are bounded.

3. Show with the following steps that the time-derivative of $V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$ is never positive:

   (a) Use the definitions for trajectory errors, equations of motion, and the control law in Equations (3.2, 2.1, 4.33) to expand $\dot{V}(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$.

   (b) Use the Jacobian relationships, Equation (4.25) and Equation (4.26), to simplify terms in $\dot{V}(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$.

   This reduces the form of $\dot{V}(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$ to the same form as in the original proof. Thus, the rest of the stability proof follows the original, with minor changes to the necessary bounds.

   This proves that $\tilde{\mathbf{x}}$ and $\dot{\tilde{\mathbf{x}}}$ tend to zero as $t \rightarrow \infty$, showing stability.

4. The Lyapunov Function is then modified to include adaptation. The application of the adaptive update law, Equation (4.35), reduces the stability proof for the adaptive algorithm once again to the nonadaptive case, where stability has been shown already.

Endpoint Lyapunov Function—Nonadaptive

The new Lyapunov Function for the system operating the nonadaptive endpoint controller is:

$$\begin{aligned} V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) &= \frac{1}{2}\dot{\tilde{\mathbf{x}}}^T \left(\mathbf{J}'^{-T}(\mathbf{q})\mathbf{M}'(\mathbf{q})\mathbf{J}'^{-1}(\mathbf{q})\right)\dot{\tilde{\mathbf{x}}} + \frac{1}{2}\tilde{\mathbf{x}}^T \left(\mathbf{K}_{Px} + c\mathbf{K}_{Vx}\right)\tilde{\mathbf{x}} \\ &\quad + c\tilde{\mathbf{x}}^T \left(\mathbf{J}'^{-T}(\mathbf{q})\mathbf{M}'(\mathbf{q})\mathbf{J}'^{-1}(\mathbf{q})\right)\dot{\tilde{\mathbf{x}}} \end{aligned} \tag{4.38}$$

The first two terms represent the "energy" in the endpoint velocity and position errors. They are squared terms, so they are always positive. The third term is a cross product term between the velocity and position errors—this term is not guaranteed to be positive, so the positive-definiteness of Equation (4.38)

such that the links lie along the same line. In this situation, the endpoint of the arm can trace out only an arc—defined by the shoulder angle and the total length of the arm—which is only one degree of freedom.

must be shown. The scalar, $c > 0$, is a weighting on the cross term—c is the same weighting that appears in the adaptive update law in Equation (3.7).

Using the inverse Jacobian relationship, Equation (4.26), and replacing $\dot{\tilde{\mathbf{x}}}$ with $\tilde{\mathbf{u}}$, the Lyapunov Function also can be expressed as[7]:

$$V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) \underset{(4.26)}{=} \frac{1}{2}\tilde{\mathbf{u}}^T \mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} + \frac{1}{2}\tilde{\mathbf{x}}^T \left(\mathbf{K}_{Px} + c\mathbf{K}_{Vx}\right)\tilde{\mathbf{x}} + c\tilde{\mathbf{x}}^T \mathbf{J}'^{-T}(\mathbf{q})\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} \tag{4.39}$$

Noting that $\mathbf{M}'(\mathbf{q})$, $\mathbf{K}_P$, and $\mathbf{K}_V$ are symmetric, the time-derivative of the Lyapunov Function can be written as:

$$\begin{aligned} \dot{V}(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) \;=\; & \tilde{\mathbf{u}}^T \mathbf{M}'(\mathbf{q})\dot{\tilde{\mathbf{u}}} + \frac{1}{2}\tilde{\mathbf{u}}^T \dot{\mathbf{M}}'(\mathbf{q}, \mathbf{u})\tilde{\mathbf{u}} + \dot{\tilde{\mathbf{x}}}^T \left(\mathbf{K}_{Px} + c\mathbf{K}_{Vx}\right)\tilde{\mathbf{x}} \\ & + c\dot{\tilde{\mathbf{x}}}^T \mathbf{J}'^{-T}(\mathbf{q})\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} + c\tilde{\mathbf{x}}^T \dot{\mathbf{J}}'^{-T}(\mathbf{q}, \mathbf{u})\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} \\ & + c\tilde{\mathbf{x}}^T \mathbf{J}'^{-T}(\mathbf{q})\dot{\mathbf{M}}'(\mathbf{q}, \mathbf{u})\tilde{\mathbf{u}} + c\tilde{\mathbf{x}}^T \mathbf{J}'^{-T}(\mathbf{q})\mathbf{M}'(\mathbf{q})\dot{\tilde{\mathbf{u}}} \end{aligned} \tag{4.40}$$

Positive-Definiteness of V

Using the first expression for $V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$ in Equation (4.38) and following the same progression as in Step 1 of the proof in Section 3.3.2 show that:

$$V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) \underset{(4.38,3.19)}{\geq} \alpha_1 \|\tilde{\mathbf{x}}\|^2 + \alpha_2 \left\|\dot{\tilde{\mathbf{x}}}\right\|^2 \tag{4.41}$$

where,

$$\begin{aligned} \alpha_1 &= \mu_p + c\mu_v - c\tfrac{\gamma_M}{\mu_J^2}\epsilon^2 \\ \alpha_2 &= \tfrac{\mu_M}{\gamma_J^2} - c\tfrac{\gamma_M}{\mu_J^2}\epsilon^{-2} \end{aligned} \tag{4.42}$$

with the following definition of the limits:

$$\begin{aligned} \mu_M &\triangleq \inf_{\mathbf{q}} \sigma_{min}\left(\mathbf{M}'(\mathbf{q})\right) \\ \mu_J &\triangleq \inf_{\mathbf{q} \notin S} \sigma_{min}\left(\mathbf{J}'(\mathbf{q})\right) \end{aligned}$$

[7]Note that:

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \dot{\mathbf{x}}_d - \dot{\mathbf{x}} \\ &= \left(\mathbf{J}'(\mathbf{q})\mathbf{u}_d + \mathbf{J}'_t(\mathbf{q})\right) - \left(\mathbf{J}'(\mathbf{q})\mathbf{u} + \mathbf{J}'_t(\mathbf{q})\right) \\ &= \mathbf{J}'(\mathbf{q})\left(\mathbf{u}_d - \mathbf{u}\right) \end{aligned}$$

such that,

$$\dot{\tilde{\mathbf{x}}} = \mathbf{J}'(\mathbf{q})\tilde{\mathbf{u}}$$

$$\begin{aligned}
\mu_p &\triangleq \sigma_{min}(\mathbf{K}_P) \\
\mu_v &\triangleq \sigma_{min}(\mathbf{K}_V) \\
\gamma_M &\triangleq \sup_{\mathbf{q}} \sigma_{max}(\mathbf{M}'(\mathbf{q})) \\
\gamma_J &\triangleq \sup_{\mathbf{q} \notin S} \sigma_{max}(\mathbf{J}'(\mathbf{q})) \\
\gamma_p &\triangleq \sigma_{max}(\mathbf{K}_P) \\
\gamma_v &\triangleq \sigma_{max}(\mathbf{K}_V)
\end{aligned}$$

S represents a set of regions around singularities. That is, the limits for $\mathbf{J}'(\mathbf{q})$ are defined for manipulator configurations that exclude finite regions around singularities. For $V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$ to be positive definite, both α_1 and α_2 must be positive, which puts a condition on c:

$$c < \frac{\mu_M \mu_J^4 \mu_v}{2\gamma_M^2 \gamma_J^2} \left(1 + \left(1 + \frac{4\mu_p \gamma_M^2 \gamma_J^2}{\mu_M \mu_J^2 \mu_v^2}\right)^{\frac{1}{2}}\right) \tag{4.43}$$

The additional terms, μ_J and γ_J, are the main changes these bounds. It is clear from Equation (4.43) that for c to have a non-trivial solution, the manipulator must remain away from singularity such that $\mu_J \neq 0$.

Boundedness for V

Following the same procedure as Step 2 of the proof in Section 3.3.2 shows that:

$$V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) \underset{(4.38,3.25)}{\leq} \beta_1 \|\tilde{\mathbf{x}}\|^2 + \beta_2 \left\|\dot{\tilde{\mathbf{x}}}\right\|^2 \tag{4.44}$$

where,

$$\begin{aligned}
\beta_1 &= \gamma_p + c\gamma_v + c\frac{\gamma_M}{\mu_J^2}\eta^2 \\
\beta_2 &= \frac{\gamma_M}{\mu_J^2} + c\frac{\gamma_M}{\mu_J^2}\eta^{-2}
\end{aligned} \tag{4.45}$$

It is again clear that if the robot remains away from singularity such that $\mu_J \neq 0$, then $V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$ remains bounded.

Negative-Semi-Definiteness of $\dot{V}$

Using the definition for trajectory error, Equation (3.2), to substitute for $\dot{\tilde{\mathbf{u}}}$, and using the equations of motion, Equation (2.1), to substitute for $\mathbf{M}'(\mathbf{q})\dot{\mathbf{u}}$ expands the Lyapunov derivative in Equation (4.40)

to:

$$\begin{aligned}\dot{V}(\tilde{\mathbf{x}},\dot{\tilde{\mathbf{x}}},t) \underset{(4.40,3.2,\ 2.1)}{=} & \mathbf{u}^T\Big(\mathbf{M}'(\mathbf{q})\dot{\mathbf{u}}_d - \mathbf{F} + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u} + \mathbf{G}'(\mathbf{q}) + \frac{1}{2}\dot{\mathbf{M}}'(\mathbf{q},\mathbf{u})\tilde{\mathbf{u}} \\ & + \mathbf{J}'^T(\mathbf{q})\mathbf{K}_{Px}\tilde{\mathbf{x}}\Big) + c\tilde{\mathbf{u}}^T\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} + c\tilde{\mathbf{x}}^T\dot{\mathbf{J}}'^{-T}(\mathbf{q},\mathbf{u})\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} \\ & + c\tilde{\mathbf{x}}^T\mathbf{J}'^{-T}\Big(\dot{\mathbf{M}}'(\mathbf{q},\mathbf{u})\tilde{\mathbf{u}} + \mathbf{M}'(\mathbf{q})\dot{\mathbf{u}}_d - \mathbf{F} + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u} + \mathbf{G}'(\mathbf{q}) \\ & + \mathbf{J}'^T\mathbf{K}_{Vx}\dot{\tilde{\mathbf{x}}}\Big) \end{aligned} \tag{4.46}$$

where the Jacobian relationship, Equation (4.25), is used for the $\mathbf{K}_{Px}$ term. Substituting the control law in Equation (4.31)—assuming an ideal model—for $\mathbf{F}$ yields:

$$\begin{aligned}\dot{V}(\tilde{\mathbf{x}},\dot{\tilde{\mathbf{x}}},t) \underset{(4.31)}{=} & uu^T\Big(-\mathbf{C}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d - \mathbf{J}'^T(\mathbf{q})\mathbf{K}_{Vx}\dot{\tilde{\mathbf{x}}} + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u} + \frac{1}{2}\dot{\mathbf{M}}'(\mathbf{q},\mathbf{u})\tilde{\mathbf{u}}\Big) \\ & + c\tilde{\mathbf{u}}^T\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} + c\tilde{\mathbf{x}}^T\dot{\mathbf{J}}'^{-T}(\mathbf{q},\mathbf{u})\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} \\ & + c\tilde{\mathbf{x}}^T\mathbf{J}'^{-T}(\mathbf{q})\Big(\dot{\mathbf{M}}'(\mathbf{q},\mathbf{u})\tilde{\mathbf{u}} - \mathbf{C}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d - \mathbf{J}'^T\mathbf{K}_{Px}\tilde{\mathbf{x}} + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u}\Big) \\ \underset{(4.25,4.26)}{=} & -c\tilde{\mathbf{x}}^T\mathbf{K}_{Px}\tilde{\mathbf{x}} - \dot{\tilde{\mathbf{x}}}^T\Big(\mathbf{K}_{Vx} - c\mathbf{J}'^{-T}(\mathbf{q})\mathbf{M}'(\mathbf{q})\mathbf{J}'^{-1}(\mathbf{q})\Big)\dot{\tilde{\mathbf{x}}} \\ & + c\tilde{\mathbf{x}}^T\dot{\mathbf{J}}'^{-T}(\mathbf{q},\mathbf{u})\mathbf{M}'(\mathbf{q})\tilde{\mathbf{u}} \\ & + \tilde{\mathbf{u}}^T\Big(-\mathbf{C}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u} + \frac{1}{2}\dot{\mathbf{M}}'(\mathbf{q},\mathbf{u})\tilde{\mathbf{u}}\Big) \\ & + c\tilde{\mathbf{x}}^T\mathbf{J}'^{-T}(\mathbf{q})\Big(-\mathbf{C}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u} + \dot{\mathbf{M}}'(\mathbf{q},\mathbf{u})\tilde{\mathbf{u}}\Big) \\ \underset{(4.11,4.26)}{=} & -c\tilde{\mathbf{x}}^T\mathbf{K}_{Px}\tilde{\mathbf{x}} - \dot{\tilde{\mathbf{x}}}^T\Big(\mathbf{K}_{Vx} - c\mathbf{J}'^{-T}(\mathbf{q})\mathbf{M}'(\mathbf{q})\mathbf{J}'^{-1}(\mathbf{q})\Big)\dot{\tilde{\mathbf{x}}} \\ & + c\tilde{\mathbf{x}}^T\dot{\mathbf{J}}'^{-T}(\mathbf{q},\mathbf{u})\mathbf{M}'(\mathbf{q})\mathbf{J}'^{-1}(\mathbf{q})\dot{\tilde{\mathbf{x}}} \\ & + \dot{\tilde{\mathbf{x}}}^T\mathbf{J}'^{-T}(\mathbf{q})\mathbf{W}^T\Big(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}}\Big) \\ & + c\tilde{\mathbf{x}}^T\mathbf{J}'^{-T}(\mathbf{q})\mathbf{W}^T\Big(-\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\dot{\tilde{\mathbf{q}}}\Big) \end{aligned} \tag{4.47}$$

The last two terms of Equation (4.47) are almost identical to those in Equation (3.36), except for the $\mathbf{J}'^{-T}(\mathbf{q})\mathbf{W}^T$ premultiplier. Assuming once again that the manipulator stays away from singularity and noting that W is a constant matrix of zeroes and ones, the development in the original proof can be used once again to state:

$$\dot{V}(\tilde{\mathbf{x}},\dot{\tilde{\mathbf{x}}},t) \underset{(3.41)}{\leq} -\lambda_1\|\tilde{\mathbf{x}}\|^2 - \lambda_2\left\|\dot{\tilde{\mathbf{x}}}\right\|^2 + \frac{1}{2}c\frac{\gamma_1}{\mu_J}\|\tilde{\mathbf{x}}\|\left\|\dot{\tilde{\mathbf{x}}}\right\|^2 \tag{4.48}$$

where

$$\gamma_4 \triangleq \sup_t |\dot{\mathbf{x}}_d| \tag{4.49}$$

$$\gamma_{j-1} \triangleq \sup_{\mathbf{q} \notin S} \sigma_{max}\left(\dot{\mathbf{J}}'^{-1}(\mathbf{q})\right) \tag{4.50}$$

$$\lambda_1 \triangleq c\left(\mu_p - \frac{\gamma_1}{\mu_J}\left(\gamma_4 + \gamma_{j-1}\right)\xi^2\right) \tag{4.51}$$

$$\lambda_2 \triangleq \left(\mu_v - c\gamma_M - \frac{\gamma_1}{\mu_J}\left(\gamma_4 + \gamma_{j-1}\right)\left(\frac{1}{2} + c\xi^{-2}\right)\right) \tag{4.52}$$

The dependence on γ_{j-1} comes from the extra term in the Lyapunov derivative involving $\dot{\mathbf{J}}'^{-T}(\mathbf{q}, \mathbf{u})$. It can be argued again that if the manipulator does not go through singularity, then $\mathbf{J}'^{-1}(\mathbf{q})$ is continuous, and thus its derivative is bounded. Requiring both λs to be positive gives the requirement:

$$\mathbf{K}_{Vx} \underset{(3.44,3.45)}{>} \frac{1}{2}\frac{\gamma_1}{\mu_J}\left(\gamma_4 + \gamma_{j-1}\right) \tag{4.53}$$

Choosing $\mathbf{K}_{Vx}$ large enough such that:

$$\lambda_2 - \frac{1}{2}c\frac{\gamma_1}{\mu_J}\left(\frac{\overline{V}}{\alpha_1}\right)^{\frac{1}{2}} > 0 \tag{4.54}$$

gives:

$$\overline{\lambda}_2 \triangleq \lambda_2 - \frac{1}{2}c\frac{\gamma 1}{\mu_J}\|\tilde{\mathbf{x}}\| > 0 \tag{4.55}$$

which yields the upper bound on the Lyapunov derivative:

$$\dot{V}(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) \underset{(4.48, 4.55)}{\leq} -\lambda_1 \|\tilde{\mathbf{x}}\|^2 - \overline{\lambda}_2 \left\|\dot{\tilde{\mathbf{x}}}\right\|^2 \leq 0 \tag{4.56}$$

Thus, the Lyapunov Function, $V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$ is *never* increasing, showing that if the initial errors are bounded, they remain bounded. Again applying Barbalat's Theorem shows that $\tilde{\mathbf{x}}$ and $\dot{\tilde{\mathbf{x}}}$ tend to zero as $t \to \infty$, proving stability.

Endpoint Lyapunov Function—Adaptive Case

The Lyapunov Function for the system utilizing the *adaptive* endpoint controller follows the example of the original proof by adding a term representing the energy in the parameter estimation errors:

$$V_1(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) = V(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) + \frac{1}{2}\tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\theta}} \tag{4.57}$$

Using the *adaptive* control law in the form of Equation (4.34) to substitute for $\mathbf{F}$ after expansion (see Equation (4.46)) and taking the time derivative of Equation (4.57) gives:

$$\dot{V}_1(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) = \dot{V}(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t) + \tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\theta}}} + \tilde{\boldsymbol{\theta}}^T \mathbf{Y}'^T(\mathbf{q}, \mathbf{u}_d, \dot{\mathbf{u}}_d, \ddot{\mathbf{u}}_d)\mathbf{J}'^{-1}(\mathbf{q})\left(\ddot{\tilde{\mathbf{x}}} + c\dot{\tilde{\mathbf{x}}}\right) \tag{4.58}$$

Noting again that $\dot{\tilde{\theta}} = -\dot{\hat{\theta}}$ and substituting the adaptive update law, Equation (4.35), eliminate the last two terms, reducing $\dot{V}_1(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$ to $\dot{V}(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, t)$. Employing the proof developed for the nonadaptive case proves the stability of the endpoint *adaptive* controller.

Chapter 5

The Task-Space Concept

The extension of the adaptive controller to endpoint control in Chapter 4 provides the essential components for further extensions to *task* space. This chapter presents the *task*-space concept in terms of a *generalized* Jacobian. It also shows that, utilizing the generalized Jacobian approach, the new endpoint adaptive control algorithm can be extended to *task* space with almost *no* modifications.

This chapter takes full advantage of the power in the generalized forces and generalized speeds concepts found in *Kane's dynamical equations* [15] to derive the expression for the generalized Jacobian. Section 5.1 presents the *task*-space adaptive control law for single-manipulator robots, showing that endpoint control is just a special case of the *task*-space adaptive control. Section 5.2 develops and formalizes the relationships between two different sets of generalized speeds chosen for a system. It also examines the relationships between the two sets of generalized active forces and equations of motion associated with the choices of generalized speeds. These generalized relationships show that the time derivative of the *task*-space control vector is equivalent to a set of generalized speeds, which leads to an intuitive validation of the *task*-space adaptive controller.

The chapter concludes with several sections providing specific examples of *task*-space control objectives:

- *Noninertial reference-frame control.* This control mode is important for mobile robots that must rely on local sensing. If the robot base rotates or accelerates during manipulation, the local sensors will be providing sensing in a noninertial frame; the controller must of course take this into consideration.

Fuel savings is another benefit of noninertial reference frame control for space robots. The controller decouples the feedback of the noninertial frame control of the manipulators from the feedback of the free-flying base control. Thus manipulator tracking errors during regulation do not cause the thrusters to fire. The thrusters will activate only in response to base regulation errors[1]. The implication is that the manipulator control gains may be increased to improve manipulator tracking without incurring unnecessary thruster firings during regulation.

- *Single-manipulator space robot control.* This example describes how *task*-space control may be utilized for control of a space robot. This example uses a single-manipulator space robot, because the *task*-space adaptive controller for multiple-manipulators has not been presented yet[2]. The characteristics of space robot control remain the same regardless of the number of manipulators.

- *Control subject to motion constraints.* Motion constraints arise when a manipulator comes into contact with its environment. Understanding the issues in control of constrained motion is critical to understanding multiple-manipulator control.

- *System-momentum control.* System-momentum control is another control mode for providing fuel saving control for space robots. Maintaining constant system momentum during local manipulation ensures that the manipulators will compensate appropriately for the reactions in the base to prevent thrusters firing, conserving precious fuel. System-momentum control has been presented by Umetani and Yoshida [44] and Koningstein [19], and this example demonstrates how *task*-space control may be used to achieve the same control mode. The added complexity of deriving symbolically the momentum equations, however, may make this control mode too costly to implement, in a given application, compared to the noninertial-reference-frame control for minimizing fuel usage[3].

- *Redundancy Management.* The *task*-space controller can be employed to control a limited class of redundant manipulator systems. The difficulty typically arises because the endpoint Jacobian is not square: There are more degrees of freedom in the robot than there are in controlled states of the end effector. Since a nonsquare Jacobian cannot be directly inverted, much of the redundancy

[1]There is full coupling in the feedforward portion of the controller.

[2]Multiple-manipulator control will be presented in Chapter 6.

[3]The complexity should be examined for each given application to determine the feasibility of implementing system-momentum control.

research has concentrated on the choice of appropriate, but computationally intensive, generalized inverses for the Jacobian.

Task-space control offers another solution by including auxiliary control objectives in the *task*-space control vector to generate a *square* generalized Jacobian. For example, the auxiliary control objective may be the height of an elbow in a 7-degree-of-freedom manipulator; the elbow height may be controlled directly to avoid obstacles while simultaneously controlling the end-effector position and orientation. Thus if suitable and valuable auxiliary control objectives can be added, *task*-space control offers a good alternative to the standard redundancy controllers.

Armed with the capabilities developed in this chapter, Chapter 6 will provide the final extension to multiple-manipulator *task*-space robot control.

5.1 Task-space Adaptive Controller—Single-Arm Case

For a robotic system, choose $\mathbf{y}^{task}$ as any set of *task*-space quantities to be controlled, physical quantities[4] that may be expressed as n linearly independent functions of the generalized coordinates, $\mathbf{q}$:

$$\mathbf{y}^{task} = \mathbf{h}(\mathbf{q}) \tag{5.1}$$

where $\mathbf{h} : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Its time derivative can be expressed as:

$$\dot{\mathbf{y}}^{task} = \mathbf{W}'(\mathbf{q})\dot{\mathbf{q}} + \mathbf{W}'_t(\mathbf{q}) \tag{5.2}$$

where $\mathbf{W}'(\mathbf{q})$ is a $n \times n$ matrix and $\mathbf{W}'_t(\mathbf{q})$ is a n vector of functions of $\mathbf{q}$ and time t[5]. Also choose for the system a set of generalized speeds, $\mathbf{u}$, related to the generalized coordinates by:

$$\mathbf{u} = \mathbf{W}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{W}_t(\mathbf{q}) \tag{5.3}$$

[4] As will be shown, these physical quantities can include not only linear translations and angular rotations, but any physical quantities that can be expressed as functions of the generalized coordinates. Keeping the arms in a high-leverage configuration and away from singularity, and minimizing thruster use are two cogent examples.

[5] In particular,

$$\mathbf{W}'(\mathbf{q}) = \frac{\partial \mathbf{h}(\mathbf{q})}{\partial \mathbf{q}}$$

and

$$\mathbf{W}'_t(\mathbf{q}) = \frac{\partial \mathbf{h}(\mathbf{q})}{\partial t}$$

where $\mathbf{W}(\mathbf{q})$ is a $n \times n$ matrix and $\mathbf{W}_t(\mathbf{q})$ is a n vector of functions of $\mathbf{q}$ and time t. Let the equations of motion for the robotic system, using this set of generalized speeds, be expressed as:

$$\mathbf{F} = \mathbf{M}'(\mathbf{q})\dot{\mathbf{u}} + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u} + \mathbf{G}'(\mathbf{q}) + \mathbf{f}_t = \mathbf{Y}'(\mathbf{q},\mathbf{u},\mathbf{u},\dot{\mathbf{u}})\boldsymbol{\theta} + \mathbf{f}_t \tag{5.4}$$

where $\mathbf{f}_t$ is added to represent the effects of prescribed motions.

Now define the *generalized* Jacobian[6] as:

$$\mathcal{J}(\mathbf{q}) \triangleq \mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q}) \tag{5.5}$$

and a time-dependent vector, $\mathcal{J}_t(\mathbf{q})$, as:

$$\mathcal{J}_t(\mathbf{q}) \triangleq \left(\mathbf{W}_t'(\mathbf{q}) - \mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\mathbf{W}_t(\mathbf{q})\right) \tag{5.6}$$

such that,

$$\dot{\mathbf{y}}^{task} = \mathcal{J}(\mathbf{q})\mathbf{u} + \mathcal{J}_t(\mathbf{q}) \tag{5.7}$$

Then the *task*-space adaptive control law is:

$$\mathbf{F} = \widehat{\mathbf{M}}'(\mathbf{q})\dot{\mathbf{u}}_d + \widehat{\mathbf{C}}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d + \widehat{\mathbf{G}}'(\mathbf{q}) + \mathbf{f}_t + \mathcal{J}^T(\mathbf{q})\left(\mathbf{K}_{Vy}\dot{\tilde{\mathbf{y}}}^{task} + \mathbf{K}_{Py}\tilde{\mathbf{y}}^{task}\right) \tag{5.8}$$

$$\tau = \mathbf{W}(\mathbf{q})^T\mathbf{F} \tag{5.9}$$

where $\tilde{\mathbf{y}}^{task} = \mathbf{y}_d^{task} - \mathbf{y}^{task}$ and $\dot{\tilde{\mathbf{y}}}^{task} = \dot{\mathbf{y}}_d^{task} - \dot{\mathbf{y}}^{task}$. The control law can be expressed in terms of the parameter vector as:

$$\mathbf{F} \underset{(2.9)}{=} \mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\widehat{\boldsymbol{\theta}} + \mathbf{f}_t + \mathcal{J}^T(\mathbf{q})\left(\mathbf{K}_{Vy}\dot{\tilde{\mathbf{y}}}^{task} + \mathbf{K}_{Py}\tilde{\mathbf{y}}^{task}\right) \tag{5.10}$$

The corresponding adaptive update law is:

$$\dot{\widehat{\boldsymbol{\theta}}} = \boldsymbol{\Gamma}\mathbf{Y}'^T(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\mathcal{J}^{-1}(\mathbf{q})\left(\dot{\tilde{\mathbf{y}}}^{task} + c\tilde{\mathbf{y}}^{task}\right) \tag{5.11}$$

In addition, the desired generalized speed and its derivative are given by:

$$\mathbf{u}_d \underset{(5.7)}{=} \mathcal{J}^{-1}(\mathbf{q})\mathbf{y}_d^{task} - \mathcal{J}^{-1}(\mathbf{q})\mathcal{J}_t(\mathbf{q}) \tag{5.12}$$

$$\dot{\mathbf{u}}_d \underset{(5.7)}{=} \mathcal{J}^{-1}(\mathbf{q})\Big(\ddot{\mathbf{y}}_d^{task} - \dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathbf{y}}^{task} - \dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\mathcal{J}^{-1}(\mathbf{q})\mathcal{J}_t(\mathbf{q}) - \dot{\mathcal{J}}_t(\mathbf{q})\Big) \tag{5.13}$$

[6]The generalized Jacobian presented in this thesis is a more formal definition than that presented by Umetani and Yoshida [44]. The generalized Jacobian is *generalized*, because it relates two sets of *generalized* speeds, and its terms obviously depend on the choice of the sets of generalized speeds.

The only requirement is that the configuration of the system must be monitored to ensure that $\mathcal{J}(\mathbf{q})$ stays away from singularity[7].

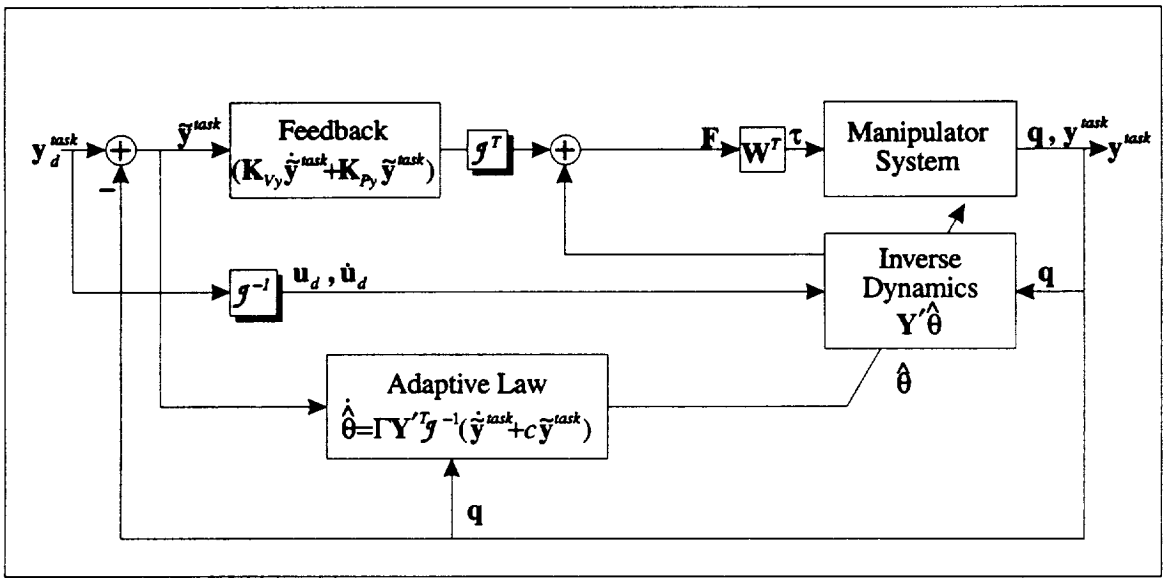


Figure 5.1: **Block Diagram of the *Task*-Space Adaptive Controller—Single-Manipulator Case**

The task-space adaptive control tracks task-space trajectories by performing feedback directly on the task-space errors, $\tilde{\mathbf{y}}^{task}$. Note that the adaptive parameter update is also based on the task-space tracking error. This controller structure is once again very similar to the original controller, Figure 3.1, and almost identical to the endpoint adaptive controller Figure 4.3. The only changes are the use of the generalized Jacobian blocks that transform the task-space quantities into equivalent generalized speeds. The transformation block, $\mathbf{W}(\mathbf{q})$, is still required to bring generalized forces into actual joint torques.

Figure 5.1 shows the structure of the *task*-space adaptive controller. Note the similarity with the endpoint adaptive controller of Figure 4.3.

In fact, comparing the *task*-space adaptive controller (Equations 5.7–5.13) with the endpoint adaptive controller (Equations 4.31–4.37) shows that choosing $\mathbf{y}^{task}$ to be the end-effector position and orientation[8], the two controllers are identical.

The *task*-space adaptive controller represents a generic controller structure, capable of handling

[7]For the general *task*-space vector—one that does not correspond only to end-effector position and orientation—the singularity points of $\mathcal{J}(\mathbf{q})$ can be caused by either *kinematic* singularity or *algorithmic* singularity. Kinematic singularity occurs when the physical configuration of the system is such that there is a loss of degree of freedom, e.g., when two links are aligned. Algorithmic singularities are all other configurations, $\mathbf{q}$, such that rank $\left(\mathcal{J}(\mathbf{q})\right) < n$.

[8]Note that although the planar examples in this thesis do not show the end-effector orientation, a general manipulator end-effector configuration is described by its position and orientation.

any number of control modes just by switching the *task*-space control mathematical vector. Switching control modes essentially involves only the switching of the generalized Jacobian blocks and the trajectory generators to produce trajectories appropriate for the new *task*-space control quantities. The feedback gains also should be changed appropriately for the controlled quantities; but none of the inverse dynamics controller, the adaptive update blocks, or the system parameters need be modified. Because the system model does not change during control mode switches, there will be no transient response in the system.

5.1.1 Proof

The stability proof for the endpoint adaptive controller in Chapter 4 *is* the stability proof for the *task*-space adaptive controller. At each step in the proof, simply replace $\mathbf{x}$ with $\mathbf{y}^{task}$, and $\mathbf{J}'(\mathbf{q})$ with $\mathcal{J}(\mathbf{q})$. All that remain are the justification for the definition of the generalized Jacobian and its relationship to *task*-space quantities.

5.2 Generalized Relationships

This section establishes the relationships between two sets of generalized speeds, partial velocities, generalized active forces, and equations of motion. Utilizing these generalized relationships, Section 5.2.5 provides additional validation of the *task*-space adaptive controller—this time on an intuitive level—showing the similarity between the *task*-space adaptive controller and the original Bayard and Wen controller.

5.2.1 Generalized Speeds

Define two sets of generalized speeds as:

$$\mathbf{u} = \mathbf{W}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{W}_t(\mathbf{q}) \tag{5.14}$$

and

$$\mathbf{u}' = \mathbf{W}'(\mathbf{q})\dot{\mathbf{q}} + \mathbf{W}'_t(\mathbf{q}) \tag{5.15}$$

Solving for $\dot{\mathbf{q}}$ in Equation (5.14) and substituting into Equation (5.15) gives[9]:

$$\mathbf{u}' = \mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\mathbf{u} + \left(\mathbf{W}'_t(\mathbf{q}) - W'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\mathbf{W}_t(\mathbf{q})\right) \tag{5.16}$$

where the terms in the parenthesis are independent of both sets of generalized speeds, $\mathbf{u}$ and $\mathbf{u}'$.

Making the generalized Jacobian definitions,

$$\mathcal{J}(\mathbf{q}) \triangleq \mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q}) \tag{5.17}$$

and a time-dependent vector, $\mathcal{J}_t(\mathbf{q})$, as:

$$\mathcal{J}_t(\mathbf{q}) \triangleq \left(\mathbf{W}'_t(\mathbf{q}) - \mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\mathbf{W}_t(\mathbf{q})\right) \tag{5.18}$$

Equation (5.16) becomes:

$$\mathbf{u}' = \mathcal{J}(\mathbf{q})\mathbf{u} + \mathcal{J}_t(\mathbf{q}) \tag{5.19}$$

Comparing $\mathbf{u}'$ in Equation (5.15) with the time-derivative of the *task*-space vector in Equation (5.2), shows that $\dot{\mathbf{y}}^{task}$ *is simply another choice of generalized speeds* corresponding to $\mathbf{u}'$. Thus, Equations (5.16–5.19) prove the generalized Jacobian relationships of Equations (5.17–5.7).

Using the generalized Jacobian notation, the relationships between the two sets of generalized speeds and their derivatives can be given by the following equations:

$$\begin{aligned}
\mathbf{u}' &= \mathcal{J}(\mathbf{q})\mathbf{u} + \mathcal{J}_t(\mathbf{q}) \\
\mathbf{u} &= \mathcal{J}^{-1}(\mathbf{q})\left(\mathbf{u}' - \mathcal{J}_t(\mathbf{q})\right) \\
\dot{\mathbf{u}} &= \mathcal{J}^{-1}(\mathbf{q})\left(\dot{\mathbf{u}}' - \dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\mathcal{J}^{-1}(\mathbf{q})\mathbf{u}' - \dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\mathcal{J}^{-1}(\mathbf{q})\mathcal{J}_t(\mathbf{q}) - \dot{\mathcal{J}}_t(\mathbf{q},\mathbf{u})\right)
\end{aligned} \tag{5.20}$$

The equations in 5.20 justify the calculations in Equation (5.12) and Equation (5.13) to derive desired generalized speeds and their time-derivatives from desired *task*-space quantities.

5.2.2 Partial Velocities

Partial angular velocities and partial velocities play large roles in the development of *Kane's dynamical equations* for generating the equations of motion of a system. This section examines the relationship between two sets of partial angular velocities and partial velocities that result from any two choices of generalized speeds.

[9] In a holonomic system, the number of generalized speeds equal the number of generalized coordinates, such that $\mathbf{W}(\mathbf{q})$ and $\mathbf{W}'(\mathbf{q})$ are both square and invertible.

The angular velocity in reference frame N of a rigid body B and the velocity in N of a point P on B can be expressed uniquely as [15]:

$$\boldsymbol{\omega} = \sum_{r=1}^{n} \boldsymbol{\omega}_r u_r + \boldsymbol{\omega}_t \tag{5.21}$$

and

$$\mathbf{v} = \sum_{r=1}^{n} \mathbf{v}_r u_r + \mathbf{v}_t \tag{5.22}$$

where $\boldsymbol{\omega}_r$, $\mathbf{v}_r$, $\boldsymbol{\omega}_t$, and $\mathbf{v}_t$ are functions[10] of $\mathbf{q}$ and time t, and u_r is the rth generalized speed. The vector, $\boldsymbol{\omega}_r$, is the rth *partial angular velocity* of B in N, and $\mathbf{v}_r$ is the rth *partial velocity*[11] of P in N. Again, when there is no prescribed motion, $\boldsymbol{\omega}_t$ and $\mathbf{v}_t$ are zero.

Rewriting Equation (5.21) and Equation (5.22) in matrix notation gives:

$$\boldsymbol{\omega} = [\boldsymbol{\omega}_r]\,\mathbf{u} + \boldsymbol{\omega}_t \tag{5.23}$$

and

$$\mathbf{v} = [\mathbf{v}_r]\,\mathbf{u} + \mathbf{v}_t \tag{5.24}$$

where the rth column of the matrix $[\boldsymbol{\omega}_r]$ contains the rth partial angular velocity, and the rth column of the matrix $[\mathbf{v}_r]$ contains the rth partial velocity[12].

Using the two sets of definitions for generalized speeds in Equation (5.14) and Equation (5.15), the expressions for $\boldsymbol{\omega}$ and $\mathbf{v}$ can be expressed as:

$$\boldsymbol{\omega} \underset{(5.23)}{=} [\boldsymbol{\omega}_r]\,\mathbf{u} + \boldsymbol{\omega}_t = [\boldsymbol{\omega}'_r]\,\mathbf{u}' + \boldsymbol{\omega}'_t \tag{5.25}$$

and

$$\mathbf{v} \underset{(5.24)}{=} [\mathbf{v}_r]\,\mathbf{u} + \mathbf{v}_t = [\mathbf{v}'_r]\,\mathbf{u}' + \mathbf{v}'_t \tag{5.26}$$

[10] Because of the dynamics of physical systems, these partial-velocity functions are *independent* of the generalized speeds.

[11] Note that the partial angular velocities *are* the partial derivatives of $\boldsymbol{\omega}$ with respect to the generalized speeds, and the partial velocities are the partial derivatives of $\mathbf{v}$ with respect to the generalized speeds. That is:

$$\boldsymbol{\omega}_r \triangleq \frac{\partial \boldsymbol{\omega}}{\partial u_r} \quad \mathbf{v}_r \triangleq \frac{\partial \mathbf{v}}{\partial u_r}$$

In practice, $\boldsymbol{\omega}_r$ and $\mathbf{v}_r$ can be derived by inspection, since they are simply the coefficients of u_r in the expression for $\boldsymbol{\omega}$ and $\mathbf{v}$, respectively.

[12] The brackets notation represents the matrix comprised of all partial velocity vectors, respectively:

$$[\boldsymbol{\omega}_r] = \begin{bmatrix} \boldsymbol{\omega}_1 & \cdots & \boldsymbol{\omega}_n \end{bmatrix} \quad [\mathbf{v}_r] = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}$$

where $[\omega'_r]$ and $[\mathbf{v}'_r]$ are the partial angular velocities and partial velocity matrices, respectively, associated with $\mathbf{u}'$.

Utilizing the relationship between the generalized speeds, substitute Equation (5.16) for $\mathbf{u}'$ into Equation (5.25) and Equation (5.26) to give:

$$\begin{aligned}[\omega_r]\,\mathbf{u} + \omega_t &= [\omega'_r]\,\mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\mathbf{u} \\ &\quad + [\omega'_r]\left(\mathbf{W}'_t(\mathbf{q}) - \mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\mathbf{W}_t(\mathbf{q})\right) + \omega'_t \end{aligned} \tag{5.27}$$

and

$$\begin{aligned}[\mathbf{v}_r]\,\mathbf{u} + \mathbf{v}_t &= [\mathbf{v}'_r]\,\mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\mathbf{u} \\ &\quad + [\mathbf{v}'_r]\left(\mathbf{W}'_t(\mathbf{q}) - \mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\mathbf{W}_t(\mathbf{q})\right)\mathbf{v}'_t \end{aligned} \tag{5.28}$$

Equating the coefficients of $\mathbf{u}$ results in the relationship between the two sets of partial velocities:

$$[\omega_r] = [\omega'_r]\,\mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q}) \underset{(5.17)}{=} [\omega'_r]\,\mathcal{J}(\mathbf{q}) \tag{5.29}$$

$$[\mathbf{v}_r] = [\mathbf{v}'_r]\,\mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q}) \underset{(5.17)}{=} [\mathbf{v}'_r]\,\mathcal{J}(\mathbf{q}) \tag{5.30}$$

These relationships will be used to relate the sets of generalized active forces associated with the two sets of generalized speeds.

5.2.3 Generalized Active Forces

This section develops the relationship between any two sets of generalized active forces. The generalized active force acting on a rigid body B is defined in Kane [15] according to the following:

If B is a rigid body belonging to a holonomic system S possessing n degrees of freedom in a reference frame N, and a set of contact and/or distance forces acting on B is equivalent to a couple of torque $\mathbf{T}_B$ together with a force $\mathbf{R}_Q$ whose line of action passes through a point Q of B, then $(F_r)_B$, the contribution of this set of forces to the generalized active force F_r for S in N is given by

$$(F_r)_B = \omega_r^B \cdot \mathbf{T}_B + \mathbf{v}_r^Q \cdot \mathbf{R}_Q \quad (r = 1, ..., n) \tag{5.31}$$

where ω_r^B and $\mathbf{v}_r^Q$ are, respectively, the rth partial angular velocity of B in N and the rth partial velocity of Q in N.

Note that the dot-products are equivalent to the following:

$$\begin{aligned} \boldsymbol{\omega}_r^B \cdot \mathbf{T}_B &= \boldsymbol{\omega}_r^{B^T} \mathbf{T}_B \\ \mathbf{v}_r^Q \cdot \mathbf{R}_Q &= \mathbf{v}_r^Q T \mathbf{R}_Q \end{aligned} \tag{5.32}$$

Stacking the contribution of B to the n generalized active forces into a column vector re-expresses Equation (5.31) in matrix notation as[13]:

$$\begin{aligned} \left[(F_r)_B\right] &\underset{(5.31)}{=} \begin{bmatrix} \boldsymbol{\omega}_1^B \cdot \mathbf{T}_B \\ \vdots \\ \boldsymbol{\omega}_n^B \cdot \mathbf{T}_B \end{bmatrix} + \begin{bmatrix} \mathbf{v}_1^Q \cdot \mathbf{R}_Q \\ \vdots \\ \mathbf{v}_n^Q \cdot \mathbf{R}_Q \end{bmatrix} \\ &\underset{(5.32)}{=} \begin{bmatrix} \boldsymbol{\omega}_1^{B^T} \mathbf{T}_B \\ \vdots \\ \boldsymbol{\omega}_n^{B^T} \mathbf{T}_B \end{bmatrix} + \begin{bmatrix} \mathbf{v}_1^Q T \mathbf{R}_Q \\ \vdots \\ \mathbf{v}_n^{Q^T} \mathbf{R}_Q \end{bmatrix} \\ &\underset{(5.25,5.26)}{=} \left[\boldsymbol{\omega}_r^B\right]^T \mathbf{T}_B + \left[\mathbf{v}_r^Q\right]^T \mathbf{R}_Q \end{aligned} \tag{5.33}$$

where the $\left[(F_r)_B\right]$ is the set of generalized active forces contributed by forces acting on body B.

Summing over all bodies in system S yields the generalized active force vector for the system:

$$\mathbf{F} \underset{(5.33)}{=} \sum_{\substack{\text{all} \\ \text{bodies}}} \left[\boldsymbol{\omega}_r^B\right]^T \mathbf{T}_B + \left[\mathbf{v}_r^Q\right]^T \mathbf{R}_Q \tag{5.34}$$

Using the alternate set of generalized speeds, $\mathbf{u}'$, the generalized forces given by Equation (5.34) becomes:

$$\mathbf{F}' \underset{(5.34,5.25,\ 5.26)}{=} \sum_{\substack{\text{all} \\ \text{bodies}}} \left[\boldsymbol{\omega}_r'^B\right]^T \mathbf{T}_B + \left[\mathbf{v}_r'^Q\right]^T \mathbf{R}_Q \tag{5.35}$$

where the primed notation indicates the generalized active forces and partial velocities associated with $\mathbf{u}'$.

To derive the relationship between $\mathbf{F}$ and $\mathbf{F}'$, substitute the partial velocity relationships of Equation (5.29) and Equation (5.30) into Equation (5.34):

$$\mathbf{F} \underset{(5.34,5.29,\ 5.30)}{=} \sum_{\substack{\text{all} \\ \text{bodies}}} \left(\left[\boldsymbol{\omega}_r'^B\right] \mathbf{W}'(\mathbf{q}) \mathbf{W}^{-1}(\mathbf{q})\right)^T \mathbf{T}_B + \left(\left[\mathbf{v}_r'^Q\right] \mathbf{W}'(\mathbf{q}) \mathbf{W}^{-1}(\mathbf{q})\right)^T \mathbf{R}_Q$$

[13]The matrix notation is defined as:

$$\left[(F_r)_B\right] \triangleq \begin{bmatrix} (F_1)_B \\ \vdots \\ (F_n)_B \end{bmatrix}$$

$$= \sum_{\substack{\text{all}\\ \text{bodies}}} \left(\mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\right)^T \left[\boldsymbol{\omega}_r'^B\right]^T \mathbf{T}_B + \left(\mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\right)^T \left[\mathbf{v}_r'^Q\right]^T \mathbf{R}_Q$$

$$= \left(\mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\right)^T \sum_{\substack{\text{all}\\ \text{bodies}}} \left[\boldsymbol{\omega}_r'^B\right]^T \mathbf{T}_B + \left[\mathbf{v}_r'^Q\right]^T \mathbf{R}_Q \tag{5.36}$$

Using Equation (5.35) as the expression for generalized forces results in the relationship:

$$\mathbf{F} \underset{(5.36,5.35)}{=} \left(\mathbf{W}'(\mathbf{q})\mathbf{W}^{-1}(\mathbf{q})\right)^T \mathbf{F}' \tag{5.37}$$

Hence, taking the definition of generalized Jacobian in Equation (5.17) gives:

$$\mathbf{F} \underset{(5.37,5.17)}{=} \mathcal{J}(\mathbf{q})^T \mathbf{F}' \tag{5.38}$$

Equation (5.38) is a *general* relationship between two sets of generalized active forces for any two sets of generalized speeds.

Utilizing the following definitions for generalized speeds of a single-manipulator robot:

$$\mathbf{u} = \dot{\mathbf{q}}$$
$$\mathbf{u}' = \mathbf{x}^{tip}$$

the corresponding generalized active forces are:

$$\mathbf{F} = \boldsymbol{\tau}$$
$$\mathbf{F}' = \mathbf{F}^{tip}$$

Substituting these into Equation (5.38) and utilizing the endpoint Jacobian gives:

$$\boldsymbol{\tau} = \mathbf{J}^T(\mathbf{q})\mathbf{F}^{tip} \tag{5.39}$$

This is the familiar relationship relating actuator torques to endpoint forces. The development in this section provides a more formalized proof of this relationship than the usual proof involving statics and virtual work[14].

5.2.4 Equations of Motion

The equations of motions for a system of rigid bodies, such as a rigid-link robot, in terms of a set of generalized speeds, $\mathbf{u}$, as:

$$\mathbf{F} = \mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{C}(\mathbf{q},\mathbf{u})\mathbf{u} + \mathbf{G}(\mathbf{q}) + \mathbf{f}_t \tag{5.40}$$

[14]See Appendix A.4 for the virtual work proof.

where the vector, $\mathbf{f}_t$, has been added to include possible forcing functions that keep the system in some prescribed motion. Using another set of generalized speeds, $\mathbf{u}'$, the equations of motion are equivalently expressed as:

$$\mathbf{F}' = \mathbf{M}'(\mathbf{q})\dot{\mathbf{u}}' + \mathbf{C}'(\mathbf{q},\mathbf{u}')\mathbf{u}' + \mathbf{G}(\mathbf{q}) + \mathbf{f}_t' \tag{5.41}$$

This section demonstrates the relationship between the terms in the two sets of equations of motion. It also demonstrates the motivation behind the choice for the *task*-space adaptive control law.

Using the generalized Jacobian relationships defined in Equation (5.20) to substitute for $\dot{\mathbf{u}}$ and $\mathbf{u}$ in the first set of equations of motion gives:

$$\begin{aligned}\mathbf{F} &= \mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathbf{u}}' + \left(\mathbf{C}(\mathbf{q},\mathbf{u}) + \mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\right)\mathcal{J}^{-1}\mathbf{u}' + \mathbf{G}(\mathbf{q}) \\ &\quad +\mathbf{f}_t - \mathbf{M}(\mathbf{q})\left(\dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\mathcal{J}^{-1}(\mathbf{q})\mathcal{J}_t(\mathbf{q}) + \dot{\mathcal{J}}_t(\mathbf{q},\mathbf{u})\right) \\ &\quad -\mathbf{C}(\mathbf{q},\mathbf{u})\mathcal{J}^{-1}(\mathbf{q})\mathcal{J}_t(\mathbf{q})\end{aligned} \tag{5.42}$$

Premultiplying Equation (5.42) by $\mathcal{J}^{-T}(\mathbf{q})$, and using Equation (5.38) to substitute $\mathbf{F}'$ for $\mathbf{F}$ gives:

$$\begin{aligned}\mathbf{F}' &= \mathcal{J}^{-T}(\mathbf{q})\mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathbf{u}}' \\ &\quad +\mathcal{J}^{-T}(\mathbf{q})\left(\mathbf{C}(\mathbf{q},\mathbf{u}) + \mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\right)\mathcal{J}^{-1}(\mathbf{q})\mathbf{u}' + \mathcal{J}^{-T}(\mathbf{q})\mathbf{G}(\mathbf{q}) \\ &\quad +\mathcal{J}^{-T}(\mathbf{q})\Big(\mathbf{f}_t - \mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\left(\dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\mathcal{J}^{-1}(\mathbf{q})\mathcal{J}_t(\mathbf{q}) + \dot{\mathcal{J}}_t(\mathbf{q},\mathbf{u})\right) \\ &\quad -\mathbf{C}(\mathbf{q},\mathbf{u})\mathcal{J}^{-1}(\mathbf{q})\mathcal{J}_t(\mathbf{q})\Big)\end{aligned} \tag{5.43}$$

When there are no prescribed motions, $\mathbf{f}_t$ and $\mathcal{J}_t(\mathbf{q})$ are zero, thus, Equation (5.43) simplifies to:

$$\begin{aligned}\mathbf{F}' &= \mathcal{J}^{-T}(\mathbf{q})\mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathbf{u}}' \\ &\quad +\mathcal{J}^{-T}(\mathbf{q})\left(\mathbf{C}(\mathbf{q},\mathbf{u}) + \mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathcal{J}}(\mathbf{q},\mathbf{u})\right)\mathcal{J}^{-1}(\mathbf{q})\mathbf{u}' \\ &\quad +\mathcal{J}^{-T}(\mathbf{q})\mathbf{G}(\mathbf{q})\end{aligned} \tag{5.44}$$

Hence, without prescribed motions, there is a simple transformation between the terms of two sets of equations of motion. Comparing terms in Equation (5.44) with the equations of motion in Equation (5.41) developed for $\mathbf{u}'$ gives:

$$\mathbf{M}'(\mathbf{q}) = \mathcal{J}^{-T}(\mathbf{q})\mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q}) \tag{5.45}$$

$$\mathbf{C}'(\mathbf{q},\mathbf{u}') = \mathcal{J}^{-T}(\mathbf{q})\left(\mathbf{C}(\mathbf{q},\mathbf{u}) + \mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathcal{J}}(\mathbf{q})\right)\mathcal{J}^{-1}(\mathbf{q}) \tag{5.46}$$

$$\mathbf{G}'(\mathbf{q}) = \mathcal{J}^{-T}(\mathbf{q})\mathbf{G}(\mathbf{q}) \tag{5.47}$$

Similarly, one can show that:

$$\mathbf{Y}'(\mathbf{q}, \mathbf{u}', \mathbf{u}', \dot{\mathbf{u}}') = \mathcal{J}^{-T}(\mathbf{q})\mathbf{Y}(\mathbf{q}, \mathbf{u}, \mathbf{u}, \dot{\mathbf{u}}) \tag{5.48}$$

where Equations (5.20) are used to express $\mathbf{u}$ and $\dot{\mathbf{u}}$ in terms of $\mathbf{u}'$ and $\dot{\mathbf{u}}'$.

Equation (5.43) and Equation (5.44) demonstrates the relationship between the equations of motion developed with *any* two sets of generalized speeds. When there are no prescribed motions, as is usual in most robotic applications, there exist direct one-to-one mappings between the two sets of inertia matrices, nonlinear matrices, and gravity torque vectors, given by Equations (5.45–5.47).

5.2.5 Task-Space Control Law Revisited

Employing the generalized relationships developed in this section, the *task*-space adaptive controller given in Section 5.1 can be written in a simple form[15]:

$$\begin{aligned} \mathbf{F}^{task} &= \widehat{\mathbf{M}}^{task}(\mathbf{q})\ddot{\mathbf{y}}_d^{task} + \widehat{\mathbf{C}}^{task}(\mathbf{q}, \dot{\mathbf{y}}_d^{task})\dot{\mathbf{y}}_d^{task} + \widehat{\mathbf{G}}^{task}(\mathbf{q}) + \mathbf{f}_t^{task} \\ &\quad + \left(\mathbf{K}_{Vy}\dot{\tilde{\mathbf{y}}}^{task} + \mathbf{K}_{Py}\tilde{\mathbf{y}}^{task}\right) \qquad (5.49) \\ \tau &= \mathbf{W}(\mathbf{q})^T\mathcal{J}^T(\mathbf{q})\mathbf{F}^{task} \qquad (5.50) \end{aligned}$$

where,

$$\begin{aligned} \mathbf{F}^{task} &= \mathcal{J}^{-T}(\mathbf{q})\mathbf{F} \\ \mathbf{M}'(\mathbf{q}) &= \mathcal{J}^{-T}(\mathbf{q})\mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q}) \\ \mathbf{C}'(\mathbf{q}, \mathbf{u}') &= \mathcal{J}^{-T}(\mathbf{q})\left(\mathbf{C}(\mathbf{q}, \mathbf{u}) + \mathbf{M}(\mathbf{q})\mathcal{J}^{-1}(\mathbf{q})\dot{\mathcal{J}}(\mathbf{q})\right)\mathcal{J}^{-1}(\mathbf{q}) \\ \mathbf{G}'(\mathbf{q}) &= \mathcal{J}^{-T}(\mathbf{q})\mathbf{G}(\mathbf{q}) \end{aligned}$$

The control law can be expressed in terms of the parameter vector as:

$$\mathbf{F}^{task} \underset{(2.9)}{=} \mathbf{Y}^{task}(\mathbf{q}, \dot{\mathbf{y}}_d^{task}, \dot{\mathbf{y}}_d^{task}, \ddot{\mathbf{y}}_d^{task})\widehat{\boldsymbol{\theta}} + \mathbf{f}_t^{task} + \left(\mathbf{K}_{Vy}\dot{\tilde{\mathbf{y}}}^{task} + \mathbf{K}_{Py}\tilde{\mathbf{y}}^{task}\right) \tag{5.51}$$

The corresponding adaptive update law is:

$$\dot{\widehat{\boldsymbol{\theta}}} = \boldsymbol{\Gamma}\mathbf{Y}^{task^T}(\mathbf{q}, \mathbf{u}_d, \mathbf{u}_d, \dot{\mathbf{u}}_d)\left(\dot{\tilde{\mathbf{y}}}^{task} + c\tilde{\mathbf{y}}^{task}\right) \tag{5.52}$$

[15] Just premultiply Equation (5.8) by $\mathcal{J}^{-T}(\mathbf{q})$.

Comparing Equations 5.49 and 5.52 with the original joint-space control law,

$$\tau = \widehat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_d + \widehat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d + \widehat{\mathbf{G}}(\mathbf{q}) + \mathbf{K}_V\dot{\tilde{\mathbf{q}}} + \mathbf{K}_P\tilde{\mathbf{q}} \tag{5.53}$$

and the parameter adaptation law,

$$\dot{\hat{\theta}} = \Gamma\mathbf{Y}^T(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)\left(\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}\right) \tag{5.54}$$

shows that the *task*-space adaptive controller is completely analogous to the original control law, except for the additional term, $\mathbf{f}_t^{task}$, to handle prescribed motions. Equations (5.49) through (5.52) show very effectively that control is performed completely in *task* space. This provides a satisfying justification for the form of the *task*-space adaptive controller, at least on an intuitive level[16].

Although the form of the adaptive controller given in Equations (5.49–5.52) *appears* much simpler than that offered in Section 5.1, it is, in general, actually much more difficult to implement. Depending on the choice of the *task*-space objectives, the terms in the equations of motion developed in *task* space, $\widehat{\mathbf{M}}^{task}(\mathbf{q})$, $\widehat{\mathbf{C}}^{task}(\mathbf{q}, \dot{\mathbf{y}}^{task})$, and $\widehat{\mathbf{G}}^{task}(\mathbf{q})$, can be *much* more complex than those derived for another set of generalized speeds. Thus, the *task*-space adaptive controller given in Section 5.1 provides maximum flexibility for implementation. One can derive the system model utilizing a set of generalized speeds that minimizes the complexity, while still able to control the system in *any* other choice of *task*-space objectives[17].

The following sections provide examples of valid *task*-space control objectives other than simple endpoint control.

5.3 Control in a Noninertial Reference Frame

Controlling in a noninertial, or accelerating, reference frame is an important control mode for a free-flying space robot[18]. This mode is useful when manipulating in the frame of the robot—for example, when performing docking or assembly while sensing both mating parts from the mobile robot base. The

[16] As Chapter 4 shows, however, the stability proof did not not carry through to *task*-space quite as readily: An additional condition that the manipulators stay away from singularity was required.

[17] This is also why this dissertation continues to distinguish between *task*-space vectors and generalized speeds, despite showing that the derivative of $\mathbf{y}^{task}$ *is* simply another set of generalized speeds.

[18] Please note that controlling in a moving or noninertial reference frame is a separate issue from simply controlling from a moving base. One can perform control from a moving base in a inertial reference frame as well as from an noninertial one. The difference is whether the desired trajectories are specified as fixed in an inertial or noninertial frame.

robot base may be rotating or otherwise accelerating, making the robot frame a noninertial reference frame.

Although all development up to this point has assumed an inertial reference frame, this section shows that control in a noninertial reference frame can still be expressed as a *task*-space objective, making direct application of the *task*-space adaptive control law feasible. This section also presents an example, based on the familiar two-link arm, that demonstrates the differences between inertial and noninertial frame control: it shows that for a space robot, the noninertial-reference-frame controller can be more fuel efficient than the inertial-frame controller.

5.3.1 Adaptive Endpoint Control in Noninertial Frame

The adaptive endpoint control in a noninertial frame also utilizes a generalized Jacobian—but one derived in the noninertial frame. The kinematic relationship, from Equation (4.15), is:

$$\mathbf{x} = \mathbf{k}(\mathbf{q}) \tag{5.55}$$

which is independent of reference frames[19]. Differentiating this relationship *is* dependent on the reference frame in which the derivative is taken. Doing so in a *noninertial* frame, B, and expressing it in terms of generalized speeds gives:

$${}^{B}\dot{\mathbf{x}} \underset{(4.23)}{=} {}^{B}\mathbf{J}'(\mathbf{q})\mathbf{u} \tag{5.56}$$

where ${}^{B}\mathbf{J}'(\mathbf{q})$ is the endpoint Jacobian with respect to the B reference frame. Following the same development as the original Jacobian yields the following inverse relationships:

$$\mathbf{u} = {}^{B}\mathbf{J}'^{-1}(\mathbf{q})\,{}^{B}\dot{\mathbf{x}} \tag{5.57}$$

$$\dot{\mathbf{u}} = {}^{B}\mathbf{J}'^{-1}(\mathbf{q})\left({}^{B}\ddot{\mathbf{x}} - {}^{B}\dot{\mathbf{J}}'(\mathbf{q},\mathbf{u})\,{}^{B}\mathbf{J}'^{-1}(\mathbf{q})\,{}^{B}\dot{\mathbf{x}}\right) \tag{5.58}$$

Thus, choosing the generalized Jacobian to be:

$$\mathcal{J}(\mathbf{q}) = {}^{B}\mathbf{J}'(\mathbf{q}) \tag{5.59}$$

allows direct application of the *task*-space control law in Section 5.1.

Note that although the noninertial Jacobian does not contain any information about the motions of the noninertial reference frame, the centrifugal and Coriolis effects of the the accelerating reference frame

[19]The *physical position* vector to the endpoint may be expressed in terms of different *coordinate systems*; but, as a *physical* vector, it is of course independent of the *reference frame*.

are still being compensated by the $\widehat{\mathbf{C}}'(\mathbf{q}, \mathbf{u}_d)\mathbf{u}_d$ and $\mathbf{f}_t$ terms in the control law (see Equation (5.8)): Those terms *do* include the motions of the reference frame.

Example

The modified two-link arm example illustrated in Figure 5.2 is used to compare the difference between inertial- and noninertial-frame control. The familiar two-link arm is placed on a mobile base with a single rotational degree of freedom, represented by q_3. The center of both coordinate systems is

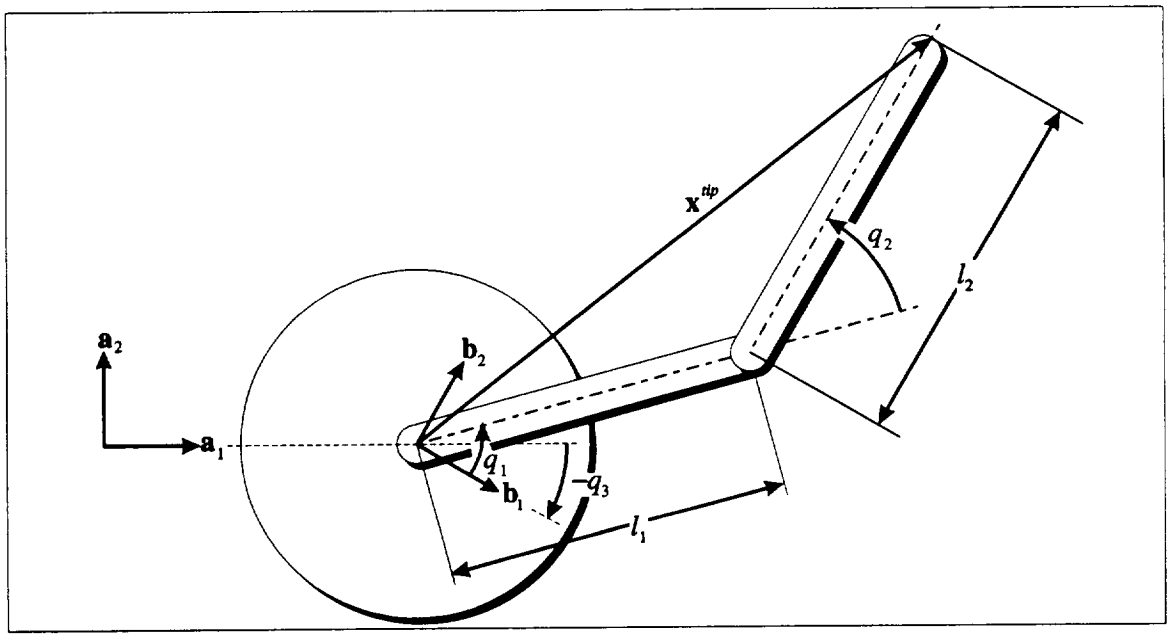


Figure 5.2: **Two-Link Arm on a Turntable**

q_1 and q_2 represent the shoulder and elbow joint angles, and q_3 is the rotation angle of the turntable. $\mathbf{x}^{tip}$ is the (invariant physical) position vector from the shoulder to the the manipulator endpoint. $\mathbf{a}_1$ and $\mathbf{a}_2$ are mutually orthogonal unit vectors fixed in the inertial frame, while $\mathbf{b}_1$ and $\mathbf{b}_2$ are mutually orthogonal unit vectors fixed in the turntable frame.

assumed to be at the shoulder. The inertial reference frame is defined by the mutually orthogonal unit vectors, $\mathbf{a}_1$ and $\mathbf{a}_2$, the rotating reference frame of the turntable base is defined by the unit vectors, $\mathbf{b}_1$ and $\mathbf{b}_2$.

Define the generalized speeds as:

$$\begin{aligned} u_1 &\triangleq \dot{q}_1 \\ u_2 &\triangleq \dot{q}_1 + \dot{q}_2 \\ u_3 &\triangleq \dot{q}_3 \end{aligned} \tag{5.60}$$

Also define $\mathbf{y}^{task}$ for inertial control as a combination of the endpoint position—expressed in the inertial coordinate system—and the base rotation:

$$\mathbf{y}^{task} \triangleq \begin{bmatrix} \mathbf{x}^{tip} \cdot \mathbf{a}_1 \\ \mathbf{x}^{tip} \cdot \mathbf{a}_2 \\ q_3 \end{bmatrix} \tag{5.61}$$

For noninertial frame control choose a combination of the same endpoint position—but expressed in the noninertial coordinate system—and base rotation:

$$\mathbf{y}^{task} \triangleq \begin{bmatrix} \mathbf{x}^{tip} \cdot \mathbf{b}_1 \\ \mathbf{x}^{tip} \cdot \mathbf{b}_2 \\ q_3 \end{bmatrix} \tag{5.62}$$

The inertial reference frame Jacobian can be expressed as:

$$\mathbf{J}'(\mathbf{q}) = \begin{bmatrix} -l_1 \sin(q_1 + q_3) & -l_2 \sin(q_1 + q_2 + q_3) & -l_1 \sin(q_1 + q_3) - l_2 \sin(q_1 + q_2 + q_3) \\ l_1 \cos(q_1 + q_3) & l_2 \cos(q_1 + q_2 + q_3) & l_1 \cos(q_1 + q_3) + l_2 \cos(q_1 + q_2 + q_3) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.63}$$

and the base-relative Jacobian is:

$${}^{B}\mathbf{J}'(\mathbf{q}) = \begin{bmatrix} -l_1 \sin(q_1) & -l_2 \sin(q_2 + q_3) & 0 \\ l_1 \cos(q_1) & l_2 \cos(q_2 + q_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.64}$$

Both controllers compensate for inertial effects, since each set of desired *task*-space motions is resolved first into equivalent desired generalized speeds and generalized-speed derivatives before incorporating them into the dynamics equations (see Equation (5.8)). The main difference appears in the feedback portion of the controllers. The last term in the control law of Equation (5.8),

$$\mathcal{J}^T(\mathbf{q}) \left(\mathbf{K}_{Vy} \dot{\tilde{\mathbf{y}}}^{task} + \mathbf{K}_{Py} \tilde{\mathbf{y}}^{task} \right) \tag{5.65}$$

illustrates the difference between the controllers.

Consider the effect of a pure endpoint error given by:

$$\tilde{\mathbf{x}} = \Delta x \mathbf{b}_1 \tag{5.66}$$

$$= \Delta x \cos(q_3)\mathbf{a}_1 + \Delta x \sin(q_3)\mathbf{a}_2 \tag{5.67}$$

Also consider, without loss of generality, that the position gains are all unity.

The feedback law of the inertial frame controller asks for a compensating generalized force given by (see Equation (5.8):

$$\mathbf{J}'(\mathbf{q})^T\tilde{\mathbf{x}} \underset{(5.63,5.67)}{=} \begin{bmatrix} -l_1 \sin(q_1) \\ -l_2 \sin(q_1 + q_2) \\ -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) \end{bmatrix} \Delta x \tag{5.68}$$

The compensating generalized force called for by the base-relative controller is (see Equation (5.8)):

$${}^B\mathbf{J}'(\mathbf{q})^T\tilde{\mathbf{x}} \underset{(5.64,5.66)}{=} \begin{bmatrix} -l_1 \sin(q_1) \\ -l_2 \sin(q_1 + q_2) \\ 0 \end{bmatrix} \Delta x \tag{5.69}$$

The first two terms in the generalized-force (mathematical) vector represent output from the shoulder and elbow motors. Equation (5.68) and Equation (5.69) show that both controllers will request the *same* amount of *arm* motor torques to compensate for the error. The *difference* is in the last term of the generalized-force vector, which represents the torque on the mobile base. Equation (5.68) shows that using the inertial reference frame Jacobian, the controller will *compensate* in the turntable for the effects of applying torques to the two-link arm. Equation (5.69) shows that the controller using the noninertial Jacobian *ignores* this effect. The base motor is actuated only in response to errors in the base orientation or angular rate, but not to endpoint-error feedback.

Although one can argue whether the latter is the desired behavior one would expect from a noninertial frame controller, this behavior is actually beneficial for space-robot control. The controller can place relatively high feedback gains on the base-relative position and velocity of the end effectors or a manipulated payload to cancel small tracking errors *without* incurring thruster firing[20], saving

[20]The thrusters *will* fire in response to base position and orientation errors.

precious fuel. The inertial-frame controller will *always* fire the thrusters[21] to compensate as needed for the feedback torques applied to the arms.

5.4 Space-Robot Control—Single Manipulator

The control of a space robot involves controlling the robot base and the manipulator simultaneously to achieve some desired trajectory of the payload, end effector or arm joints. For payload control, a natural *task*-space control mathematical vector is simply the one containing both the payload and robot base positions and orientations. Additionally, the payload motions can be expressed in either an inertial reference frame or the base-relative frame.

Figure 5.3 shows an example utilizing a planar, one-arm space robot. The base has three degrees of freedom—two in translation and one in rotation—and the arm has two degrees of freedom, totalling five for the system, as represented by q_1 through q_5.

For endpoint control in the inertial reference frame choose[22]:

$$\mathbf{y}^{task} = \begin{bmatrix} \mathbf{x}^{OQ} \cdot \mathbf{a}_1 \\ \mathbf{x}^{OQ} \cdot \mathbf{a}_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} \tag{5.70}$$

such that,

$$\dot{\mathbf{y}}^{task} = \begin{bmatrix} {}^{A}\dot{\mathbf{x}}^{Q} \cdot \mathbf{a}_1 \\ {}^{A}\dot{\mathbf{x}}^{Q} \cdot \mathbf{a}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \end{bmatrix} \tag{5.71}$$

Implementation of *task*-space adaptive controller, therefore, will provide adaptation while simultaneously controlling the manipulator endpoint position and the robot base position and orientation.

[21] Actually, the thrusters will not fire unless the compensating forces are above the threshold of the on/off thruster controller. This, however, prevents the use of higher gains for the control of endpoint or payload positions.

[22] For the special case that the end effector grasps the object at its center of mass, endpoint control is the same as object control.

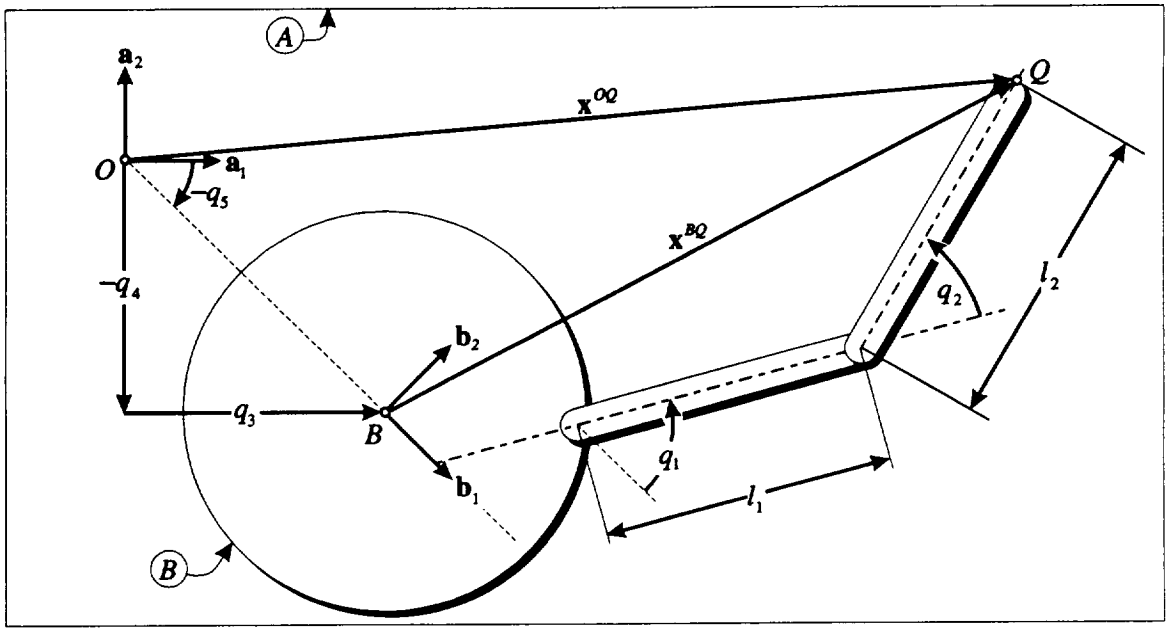


Figure 5.3: **One-Arm Space Robot**

q_1 and q_2 represent the shoulder and elbow joint angles. q_3 and q_4 represent the position of point B on the robot base, resolved into components along $\mathbf{a}_1$ and $\mathbf{a}_2$, respectively. q_5 is the orientation of the robot base. $\mathbf{x}^{OQ}$ is the position vector from the center of the coordinate system fixed in A to the manipulator endpoint, while $\mathbf{x}^{BQ}$ is the position vector from the center of the robot base to the endpoint. $\mathbf{a}_1$ and $\mathbf{a}_2$ are mutually orthogonal unit vectors fixed in the inertial frame, while $\mathbf{b}_1$ and $\mathbf{b}_2$ are mutually orthogonal unit vectors fixed in (painted on) the robot base frame.

On the other hand, for endpoint control in a useful combination of frames, we can choose the *task*-space vector to be:

$$\mathbf{y}^{task} = \begin{bmatrix} \mathbf{x}^{BQ} \cdot \mathbf{b}_1 \\ \mathbf{x}^{BQ} \cdot \mathbf{b}_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} \tag{5.72}$$

such that,

$$\dot{\mathbf{y}}^{task} = \begin{bmatrix} {}^{B}\dot{\mathbf{x}}^{Q} \cdot \mathbf{b}_1 \\ {}^{B}\dot{\mathbf{x}}^{Q} \cdot \mathbf{b}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \end{bmatrix} \tag{5.73}$$

which generates a controller that simultaneously controls the endpoint in the *base frame* and the robot base in the *inertial frame*. A significant benefit of the base-relative controller, as mentioned in Section 5.3, is that higher feedback gains can be used on the endpoint tracking errors without incurring excessive thruster firings, thereby saving fuel.

5.5 Control with Motion Constraints

Motion constraints occur when the manipulator comes into contact with a surface or another object. These constraints can often be incorporated into the *task*-space vector by considering the *relative* velocity between the endpoint and the surface or payload object. Figure 5.4 shows a planar arm about to grasp an object. The payload is modelled as part of the system, where its position is described by q_3 and q_4. The object is assumed, for simplicity of presentation, to be circularly symmetric, and the grasp point, $\mathbf{x}^{OQ'}$, is at the center of the object[23].

For this situation it is advantageous to let the *task*-space vector be composed of the endpoint position and the *relative* position between the endpoint and the grasp point on the object:

$$\mathbf{y}^{task} = \begin{bmatrix} \mathbf{x}^{OQ} \cdot \mathbf{a}_1 \\ \mathbf{x}^{OQ} \cdot \mathbf{a}_2 \\ \left(\mathbf{x}^{OQ} - \mathbf{x}^{OQ'}\right) \cdot \mathbf{a}_1 \\ \left(\mathbf{x}^{OQ} - \mathbf{x}^{OQ'}\right) \cdot \mathbf{a}_1 \end{bmatrix} \tag{5.74}$$

[23]The only reason for this requirement is because, with only two manipulator joints, and no orientation control at the end effector, the orientation of the object is uncontrollable. The problem can be expanded readily to include a rotational joint at the gripper.

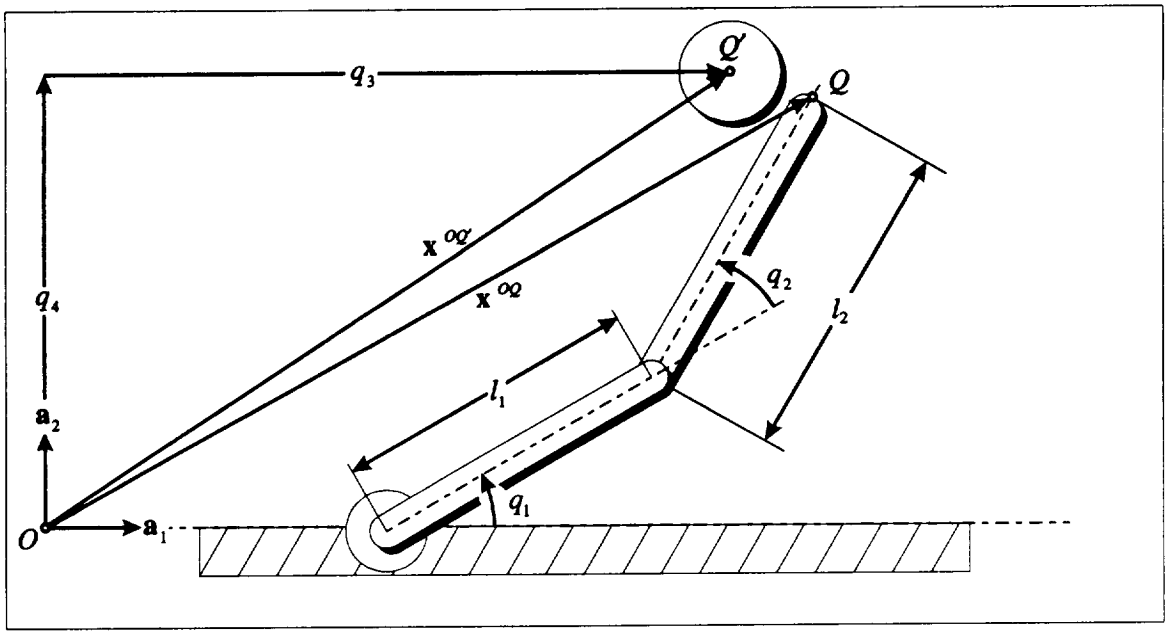


Figure 5.4: **Motion Constraint**

q_1 and q_2 represent the shoulder and elbow joint angles. The object is modelled as part of the system, and its position is given by q_3 and q_4. Q marks the endpoint of the manipulator, and Q' is the grasp point on the object. When the manipulator grasps the object, Q coincides with Q'. $\mathbf{a}_1$ and $\mathbf{a}_2$ are mutually orthogonal unit vectors fixed in the inertial frame.

such that

$$\dot{\mathbf{y}}^{task} = \begin{bmatrix} \dot{\mathbf{x}}^{Q} \cdot \mathbf{a}_1 \\ \dot{\mathbf{x}}^{Q} \cdot \mathbf{a}_2 \\ \left(\dot{\mathbf{x}}^{Q} - \dot{\mathbf{x}}^{Q'}\right) \cdot \mathbf{a}_1 \\ \left(\dot{\mathbf{x}}^{Q} - \dot{\mathbf{x}}^{Q'}\right) \cdot \mathbf{a}_1 \end{bmatrix} \tag{5.75}$$

$$= \begin{bmatrix} \mathbf{J}(\mathbf{q}) & \mathbf{0} \\ \mathbf{J}(\mathbf{q}) & -\mathbf{I}_{2\times 2} \end{bmatrix} \mathbf{u} \tag{5.76}$$

$$= \mathcal{J}(\mathbf{q})\mathbf{q} \tag{5.77}$$

where $\mathbf{J}(\mathbf{q})$ is the endpoint Jacobian, $\mathbf{I}_{2\times 2}$ is the 2×2 identity matrix, $\mathbf{u} = \dot{\mathbf{q}}$, and $\mathcal{J}(\mathbf{q})$ is the resulting generalized Jacobian. To enforce the motion constraints, the controller simply specifies that the last two terms of the desired trajectories, $\mathbf{y}_d^{task}$, $\dot{\mathbf{y}}_d^{task}$, and $\ddot{\mathbf{y}}_d^{task}$, be zero.

This example will be useful when formulating multiple-manipulator cooperative control in Chapter 6. The force constraints that occur at the grasp point will also be examined in Chapter 6.

5.6 System-Momentum Control

Control of the total system momentum can be useful for space robots as a fuel savings measure. By controlling endpoint or object motions while maintaining constant system momentum, no thrusters will be fired, ensuring that no fuel is expended. With no thrusters, the robot base will of course react to the motions of the manipulators. The controller compensates appropriately for these base motions to allow accurate trajectory tracking of the endpoint or payload object.

This section shows that the system linear momentum and angular momentum can be expressed as yet another set of *task*-space objectives. In particular, they are shown to be a linear function of generalized speeds[24].

The linear momentum $\mathbf{L}^S$ of a system of p bodies is the sum of the linear momentum of each body i in the system[25]:

$$\mathbf{L}^S = \sum_{i=1}^{p} \mathbf{L}^i \tag{5.78}$$

$$= \sum_{i=1}^{p} m_i \mathbf{v}^{i^*} \tag{5.79}$$

where $\mathbf{v}^{i^*}$ is the velocity of the center of mass of body i.

Expressed in terms of generalized speeds, $\mathbf{L}^S$ becomes[26]:

$$\mathbf{L}^S = \sum_{i=1}^{p} m_i \sum_{r=1}^{n} \mathbf{v}_r^{i^*} u_r \tag{5.80}$$

$$= \sum_{i=1}^{p} \sum_{r=1}^{n} m_i \mathbf{v}_r^{i^*} u_r \tag{5.81}$$

$$= \sum_{r=1}^{n} \sum_{i=1}^{p} m_i \mathbf{v}_r^{i^*} u_r \tag{5.82}$$

$$= \sum_{r=1}^{n} \mathbf{L}_r^S u_r \tag{5.83}$$

$$= \left[\mathbf{L}_r^S\right] \mathbf{u} \tag{5.84}$$

where $\mathbf{L}_r^S$ is the *partial linear momentum* given by:

$$\mathbf{L}_r^S = \sum_{i=1}^{p} m_i \mathbf{v}_r^{i^*} \tag{5.85}$$

[24] See Koningstein [19] for more details.

[25] Formally, this is the linear momentum of S in frame A if the velocities are also in A.

[26] Because the whole system is being modelled, there are no prescribed motions, making $\mathbf{v}_t^{i^*}$ zero.

and the matrix notation, $\left[\mathbf{L}_r^S\right]$, is given by:

$$\left[\mathbf{L}_r^S\right] = \left[\begin{array}{ccc} \mathbf{L}_1^S & \cdots & \mathbf{L}_n^S \end{array}\right] \tag{5.86}$$

Similar arguments follow for the central angular momentum $\mathbf{H}^{S/S^*}$ of a system of p bodies about the system mass center S^*:

$$\begin{aligned}
\mathbf{H}^{S/S^*} &= \sum_{i=1}^{p} \mathbf{H}^{i/i^*} + \mathbf{H}^{i/S^*} && (5.87)\\
&= \sum_{i=1}^{p} \mathbf{I}^{i/i^*}\boldsymbol{\omega}^i + \mathbf{x}^{S^*i^*} \times m_i \mathbf{v}^{i^*} && (5.88)\\
&= \sum_{i=1}^{p} \mathbf{I}^{i/i^*} \sum_{r=1}^{n} \boldsymbol{\omega}_r^i u_r + \mathbf{x}^{S^*i^*} \times m_i \sum_{r=1}^{n} \mathbf{v}_r^{i^*} u_r && (5.89)\\
&= \sum_{r=1}^{n} \left(\sum_{i=1}^{p} \mathbf{I}^{i/i^*}\boldsymbol{\omega}_r^i + \mathbf{x}^{S^*i^*} \times m_i \mathbf{v}_r^{i^*} \right) u_r && (5.90)\\
&= \sum_{r=1}^{n} \left(\sum_{i=1}^{p} \mathbf{H}_r^{i/i^*} + \mathbf{x}^{S^*i^*} \times \mathbf{L}_r^i \right) u_r && (5.91)\\
&= \sum_{r=1}^{n} \mathbf{H}_r^{S/S^*} u_r && (5.92)\\
&= \left[\mathbf{H}_r^{S/S^*}\right] \mathbf{u} && (5.93)
\end{aligned}$$

where $\mathbf{x}^{S^*i^*}$ is the position vector from the system center of mass to the center of mass of body i, and $\mathbf{H}_r^{S/S^*}$ is the *partial central angular momentum* of the system given by:

$$\mathbf{H}_r^{S/S^*} = \sum_{i=1}^{p} \mathbf{H}_r^{i/i^*} + \mathbf{x}^{S^*i^*} \times \mathbf{L}_r^i \tag{5.94}$$

The matrix notation, $\left[\mathbf{H}_r^{S/S^*}\right]$, is given by:

$$\left[\mathbf{H}_r^{S/S^*}\right] = \left[\begin{array}{ccc} \mathbf{H}_1^{S/S^*} & \cdots & \mathbf{H}_n^{S/S^*} \end{array}\right] \tag{5.95}$$

Thus both $\mathbf{L}^S$ and $\mathbf{H}^{S/S^*}$ can be viewed as alternate generalized speeds (see Equation (5.15)). As such, they are valid choices for elements in the *task*-space vector[27].

When no adaptation is required, the *task*-space controller presented in Section 5.1 can be used directly to control system momentum. That controller, however, cannot in theory be implemented for

[27] Since the *task*-space vector needs to have n elements—matching the degrees of freedom in the system—the system linear momentum and angular momentum by themselves do not of course provide sufficient degrees of freedom to complete a *task*-space mathematical vector.

adaptation, because the terms in the generalized Jacobian corresponding to $\left[\mathbf{L}_r^S\right]$ and $\left[\mathbf{H}_r^{S/S^*}\right]$ contain *unknown* parameters, such as masses, moments of inertia, and the center-of-mass location for each body i. In this case, the adaptive controller formulated completely in *task* space, as given in Section 5.2.5, may be required. Utilizing this form of the controller, however, increases controller complexity. The noninertial reference frame controller may be a better candidate for fuel-efficient control.

The one-arm space robot illustrates the control of system momentum. Figure 5.5 depicts the system. Since this is a planar system, the system linear momentum involves two degrees of freedom in the plane,

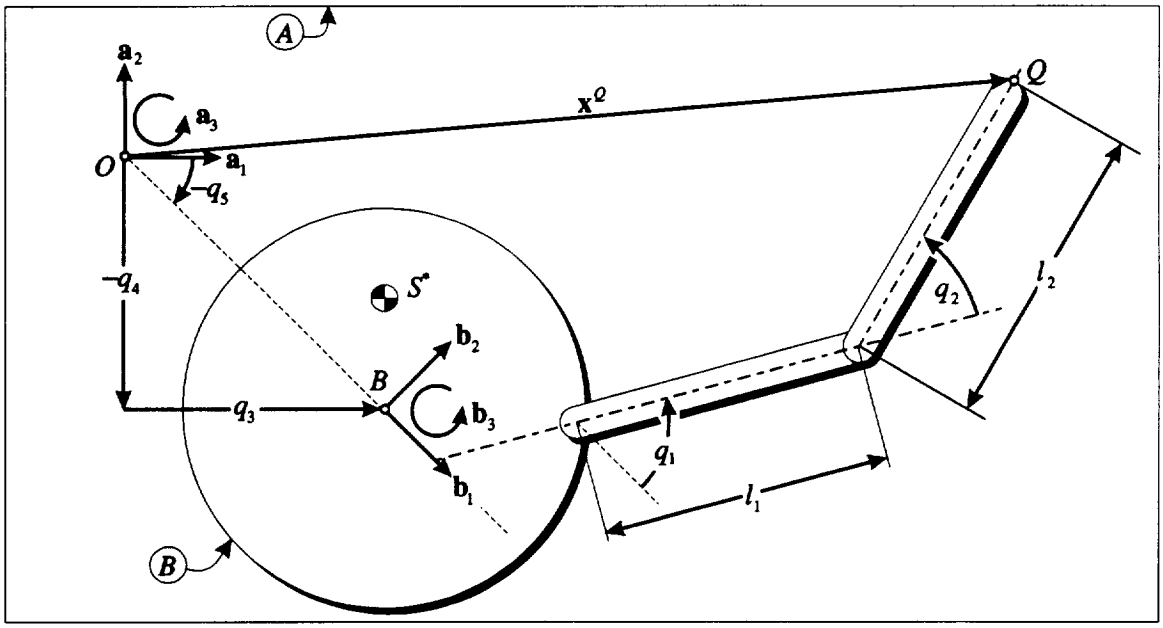


Figure 5.5: One-Arm Space Robot—Momentum Control

q_1 and q_2 represent the shoulder and elbow joint angles. q_3, q_4, and q_5 represent the position and orientation of the robot base. Q marks the manipulator endpoint, and S^* marks the system mass center at this instant. $\mathbf{a}_1$ and $\mathbf{a}_2$ are mutually orthogonal unit vectors fixed in the inertial frame, while $\mathbf{b}_1$ and $\mathbf{b}_2$ are mutually orthogonal unit vectors fixed in the robot base frame. $\mathbf{a}_3$ and $\mathbf{b}_3$ are unit vectors perpendicular to the plane and point out from the plane.

and the angular momentum only involves one degree of freedom—rotation about an axis in a direction

perpendicular to the plane. Choose the *task*-space mathematical vector such that its derivative is:

$$\dot{\mathbf{y}}^{task} = \begin{bmatrix} {}^{A}\dot{\mathbf{x}}^{Q} \cdot \mathbf{a}_1 \\ {}^{A}\dot{\mathbf{x}}^{Q} \cdot \mathbf{a}_2 \\ \mathbf{L}^{S} \cdot \mathbf{a}_1 \\ \mathbf{L}^{S} \cdot \mathbf{a}_2 \\ \mathbf{H}^{S/S^*} \cdot \mathbf{a}_3 \end{bmatrix} \tag{5.96}$$

$$= \begin{bmatrix} \mathbf{J}(\mathbf{q}) \\ \left[\mathbf{L}_r^{S}\right] \\ \left[\mathbf{H}_r^{S/S^*}\right] \end{bmatrix} \mathbf{u} \tag{5.97}$$

$$= \mathcal{J}(\mathbf{q})\mathbf{u} \tag{5.98}$$

where $\mathbf{J}(\mathbf{q})$ is the 2×5 endpoint Jacobian in the inertial frame, $\left[\mathbf{L}_r^{S}\right]$ is the 2×5 partial linear momentum matrix, and $\left[\mathbf{H}_r^{S/S^*}\right]$ is the 1×5 partial angular momentum matrix.

To maintain zero system momentum while controlling the endpoint, simply set the last three terms in the desired *task*-space trajectories, $\mathbf{y}_d^{task}$, $\dot{\mathbf{y}}_d^{task}$, and $\ddot{\mathbf{y}}_d^{task}$, to zero.

Base-relative endpoint control can be achieved by choosing to use the base-relative endpoint Jacobian, ${}^{B}\mathbf{J}(\mathbf{q})$, in $\dot{\mathbf{y}}^{task}$[28].

5.7 Redundancy Management

Task-space control offers a limited solution to the control of redundant mechanisms. Whether a system is redundant involves consideration of the number of degrees of freedom (DOF) in the system versus the number of degrees of freedom to be controlled. If the number of controlled degrees of freedom is less than that of the total system, the control is said to be redundant.

The classic example is the control of a manipulator end effector in six degrees of freedom—three in position and three in orientation—with a 7 DOF manipulator. The endpoint Jacobian for this system is nonsquare, so the direct matrix inverse used to derive joint rates from endpoint velocities is unavailable. Most redundant-control research has concentrated on utilizing the generalized inverse of the Jacobian:

$$\dot{\mathbf{q}}_d = \mathbf{J}^T(\mathbf{q}) \left(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q})\right)^{-1} \dot{\mathbf{x}}_d + \left(\mathbf{I}_{n\times n} - \mathbf{J}^T(\mathbf{q}) \left(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q})\right)^{-1} \mathbf{J}(\mathbf{q})\right) \dot{\mathbf{x}}_t \tag{5.99}$$

[28] In actual implementation, proportional or pulse-width-modulation thruster control would still use a significant amount of fuel, because the base-relative endpoint errors, while small, do not stay exactly zero. Combining this control mode with on-off thruster control, on the other hand, would keep fuel use zero for extended periods.

where $\dot{\mathbf{x}}_t$ can be any vector. The last term represents motion in the null space of $\mathbf{J}(\mathbf{q})$; that is, internal motion that will not affect the endpoint motion. Without $\dot{\mathbf{x}}_t$, the desired set of joint rates $\dot{\mathbf{q}}_d$ minimizes the energy expended in meeting the desired endpoint trajectory $\dot{\mathbf{x}}_d$. Many researchers argue of course for the attractive option that adding $\dot{\mathbf{x}}_t$ can make the manipulator achieve additional objectives, such as singularity avoidance or obstacle avoidance, while tracking endpoint trajectories. In practice, however, it often occurs that choosing $\dot{\mathbf{x}}_t$ to achieve the additional objectives is difficult. Moreover, solving the generalized Jacobian inverse, $\mathbf{J}^T(\mathbf{q})\left(\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q})\right)^{-1}$, can be computationally expensive.

The *task*-space control approach, however, allows *direct* choice of the additional objectives. In the one-arm space robot example, the additional objectives are represented by either the direct control of the robot base motions or the system momentum[29]. For those tasks involving redundant manipulators where additional control objects can be appropriately chosen, *task*-space control offers a simple solution.

5.8 Conclusions

Task-space control is a powerful concept for use in the control of robotic systems. It allows the control engineer to develop the system model, or equations of motion, in its simplest form utilizing an appropriate set of generalized speeds; yet it allows him to control *any* variables that can be expressed as functions of the generalized coordinates. The *task*-space control vector can contain a *mixture* of physically dissimilar quantities, containing joint motions, Cartesian motions, motion constraints, and even system momenta, as long as the size of the *task*-space vector matches the total number of degrees of freedom of the robotic system. Finally, utilizing the stability proof for the endpoint adaptive controller, one can show with negligible effort that the same proof demonstrates the stability of the *task*-space adaptive controller.

[29]This is also the approach chosen by Seraji [35] for the control of single-manipulator robots.

Chapter 6

System Concatenation

Multiple manipulators grasping a common object form closed kinematic chains. Given the fully reduced set of equations of motion for such a system, the new *task*-space controller may be used directly for control. The closed-kinematic-chain equations of motion, however, can be very complex to derive, hampering implementation. *System concatenation* is a concept that eliminates the requirement for deriving the complex closed-chain equations, but can still work well with the new *task*-space adaptive controller to provide cooperative control for multiple manipulators.

System concatenation is a modelling technique that provides for efficient, incremental generation of *total* system models for multiple, interacting system, such as multiple arms cooperatively manipulating an object. It takes full advantage of models already developed for each manipulator or robot subsystem to minimize the effort in deriving the total system models. This formulation eliminates the need to develop and solve for the closed-kinematic-chain equations of motion, simplifying implementation. It also keeps separated the parameters of each robot subsystem, allowing the ability to separately "tune" the adaptive control for each subsystem.

Essentially, *system concatenation* simply stacks, or concatenates, the equations of motion of each subsystem into a larger, total system model. The *coupling* between each subsystem is handled through constraints—motion constraints and force constraints. This idea is not new. Many simulation techniques are based on this concept. Its use in *controls* in its generalized form, however, is to the author's knowledge, novel. *System concatenation* does of course increase the order of the set of system equations. This disadvantage, however, is offset by the ease of forming the equations of motion, minimizing the possibilities for error when the algorithm is implemented on a digital computer.

The *system concatenation* approach utilizes *task*-space control to incorporate the motion constraints into the *task*-space control objective, and specifies desired *task*-space trajectories that are consistent with these constraints. Explicit modelling of the force constraints also allows the controller to specify load distribution amongst the manipulators in addition to specifying internal forces to be applied on the manipulated payload.

Section 6.1 defines the structure of the equations of motion for the *concatenated* system. Sections 6.2 and 6.3 derive the motion and force constraints in terms of the generalized Jacobian, allowing them to be incorporated into the *task*-space controller. Section 6.4 utilizes the constraint forces to derive the mapping from object forces and torques to end-effector forces and torques—a mapping that allows the specification of load distribution and of the "squeeze" forces experienced by the object being manipulated.

6.1 Modelling

A pair of manipulators grasping a common object form a closed kinematic chain. The motion constraints at the grasp points reduce the number of degrees of freedom for the system. The equations of motion for the reduced-order system involve solving the closed-kinematic equations—a very complex set of equations in general. Moreover, to perform adaptive control, these equations must be solved symbolically in order to determine how the equations of motion depend on the physical parameters of the system. While this can be done for any system, updating the model as the system changes—adding manipulators, for example—can be extremely costly.

As Koningstein [19] demonstrated at ARL, however, a well-designed controller did not require the reduced order equations of motion. The controller treated the system as an open-chain system by breaking the kinematic chain at a grasp point and modelling the payload as an additional link on one of the manipulators. The controller continued to enforce the motion and force constraints at the grasp point by specifying desired trajectories and desired forces that were consistent with the constraints. Doing so allowed the manipulator system to perform its task without putting undue stress on the manipulated object.

System concatenation carries this approach a further step. It breaks the kinematic chains at *all* grasp points[1]. One benefit is that the controller can take advantage of the system models already developed

[1] If there are more than two manipulators there are more than one kinematic chain.

for individual manipulators and objects. It does not have to modify the equations of motion of a manipulator to include the object as an additional link. All manipulators are treated equivalently. Most importantly, this approach is applicable for any number of manipulators and objects.

Another benefit of *system concatenation* for adaptive control is that the physical parameters of individual manipulators and objects are kept separated. This allows the adaptive controller to isolate the physical parameters associated with each subsystem into separate portions of the adaptable parameter vector. If only a subsystem, such as the payload object, is unknown, the parameter separation property minimizes the number of parameters that must be adapted. The parameter separation property also ensures that the controller can take full advantage of all that *are* known in the system. At the same time, separable parameters does not prevent the adaptive controller from simultaneously adaptive to all the parameters for the entire system.

6.1.1 System Concatenation Formulation

System concatenation considers individual robots and objects as subsystems of the entire manipulation system. Let each of these subsystems i be modelled as:

$$\mathbf{F}_i = \mathbf{M}'_i(\mathbf{q}_i)\dot{\mathbf{u}}_i + \mathbf{C}'_i(\mathbf{q}_i, \mathbf{u}_i)\mathbf{u}_i + \mathbf{G}'_i(\mathbf{q}_i) = \mathbf{Y}'_i(\mathbf{q}_i, \mathbf{u}_i, \mathbf{u}_i, \dot{\mathbf{u}}_i)\boldsymbol{\theta}_i \tag{6.1}$$

where $\mathbf{q}_i \in \mathbb{R}^{n_i}$ is a vector of generalized coordinates[2], $\mathbf{u}_i \in \mathbb{R}^{n_i}$ is a vector of generalized speeds, $\mathbf{F}_i \in \mathbb{R}^{n_i}$ is the vector of generalized active forces, $\mathbf{M}'_i(\mathbf{q}_i) \in \mathbb{R}^{n_i \times n_i}$ is the symmetric, positive-definite inertia matrix, $\mathbf{C}'_i(\mathbf{q}_i, \mathbf{u}_i) \in \mathbb{R}^{n_i \times n_i}$ is the matrix of Coriolis and centrifugal terms, $\mathbf{G}'_i(\mathbf{q}_i) \in \mathbb{R}^{n_i}$ is the vector of gravity torques, $\mathbf{Y}'_i(\mathbf{q}_i, \mathbf{u}_i, \mathbf{u}_i, \dot{\mathbf{u}}_i) \in \mathbb{R}^{n_i \times p_i}$ is the regressor[3], and $\boldsymbol{\theta}_i \in \mathbb{R}^{p_i}$ is the parameter vector, all defined for the subsystem i. The equations of motion for an entire system with m subsystems can be expressed as:

$$\begin{bmatrix} \mathbf{F}_1 \\ \vdots \\ \mathbf{F}_m \end{bmatrix} + \mathbf{F}_C = \begin{bmatrix} \mathbf{M}'_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M}'_m \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_1 \\ \vdots \\ \dot{\mathbf{u}}_m \end{bmatrix} + \begin{bmatrix} \mathbf{C}'_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}'_m \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_m \end{bmatrix} \tag{6.2}$$

[2]Generalized coordinates are used in the development for maintaining mathematical generality to facilitate implementation on a digital computer. In all the examples, the generalized coordinates are simply chosen to be convenient geometric coordinates that can be easily measured.

[3]The term "regressor" is typically used in system identification to refer to the mathematical matrix that contains the states of the system and none of the plant parameters.

$$+\begin{bmatrix} \mathbf{G'}_1 \\ \vdots \\ \mathbf{G'}_m \end{bmatrix} \tag{6.3}$$

$$\mathbf{0} = \dot{\mathbf{x}}_C \tag{6.4}$$

where $\dot{\mathbf{x}}_C \in \mathbb{R}^N$ represents N motion constraints, $\mathbf{F}_C \in \mathbb{R}^n$ is vector of generalized constraint forces that result from the motion constraints, and $n = n_1 + \ldots + n_m$ is the degrees of freedom for the entire system. The equations of motion can be expressed in terms of the regressor and parameter vectors as:

$$\begin{bmatrix} \mathbf{F}_1 \\ \vdots \\ \mathbf{F}_m \end{bmatrix} + \mathbf{F}_C = \begin{bmatrix} \mathbf{Y'}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Y'}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_m \end{bmatrix} \tag{6.5}$$

$$\mathbf{0} = \dot{\mathbf{x}}_C \tag{6.6}$$

Note how the individual parameter sets, $\boldsymbol{\theta}_i$, are kept separated.

This demonstrates that the equations of motion for the entire system is almost trivial to form, given the individual equations of motion. What remains are the examinations of the motion constraints and the generalized constraint forces.

6.1.2 Example

The example in Figure 6.1 illustrates the *system concatenation* approach[4]. The picture at the left represents the separate subsystems and the one at the right represents the entire system when connected. The individual equations of motion, ignoring gravity, are listed with each subsystem. When not grasping the payload, the system has seven (7) degrees of freedom. When grasping, the constraints reduce the system to three (3) degrees of freedom, most easily seen as the position and orientation of the object. That is, given the position and orientation of the object, all the arm joint angles are uniquely determined.

The generalized coordinates, $\mathbf{q}_{arm_1} \in \mathbb{R}^2$ and $\mathbf{q}_{arm_2} \in \mathbb{R}^2$, correspond with the joint angles for the two manipulators, and $\mathbf{q}_{obj} \in \mathbb{R}^3$ represents the position of P—the center of mass of the

[4]The example shows that *system concatenation* simply utilizes D'Alembert's Principle to combine the equations *already* developed for the individual systems to form the full system equations of motion. One could have started with D'Alembert's Principle, of course, to derive the full equations of motion from scratch.

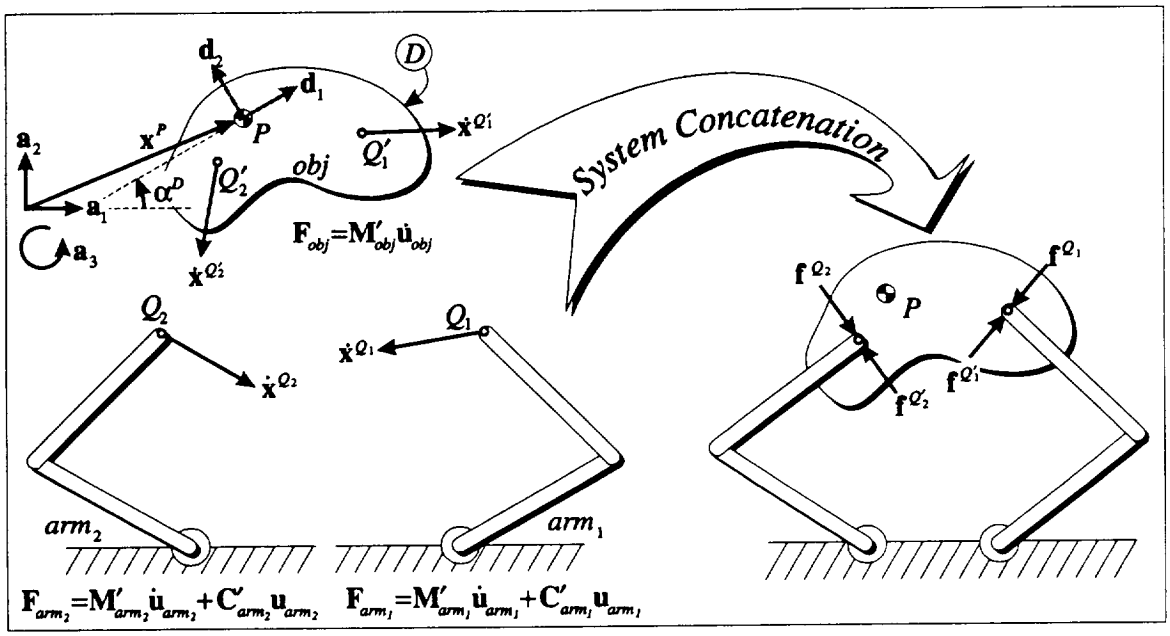


Figure 6.1: *System Concatenation*

The system of two manipulators and the not-yet-captured payload object is illustrated at the left as separate subsystems. Their individual equations of motion are listed below each subsystem. System concatenation *brings the separate subsystems together to model the complete system after the object is grasped.* $\dot{\mathbf{x}}^{Q_1}$ *and* $\dot{\mathbf{x}}^{Q_2}$ *are the endpoint velocities,* $\dot{\mathbf{x}}^{Q_1'}$ *and* $\dot{\mathbf{x}}^{Q_2'}$ *are the velocities of the grasp points.* $\mathbf{f}^{Q_1}$ *and* $\mathbf{f}^{Q_2}$ *represent forces on each manipulator at its endpoint, and* $\mathbf{f}^{Q_1'}$ *and* $\mathbf{f}^{Q_2'}$ *represent forces on the object at each grasp point.*

payload—and the orientation of payload body D:

$$\mathbf{q}_{obj} = \begin{bmatrix} \mathbf{x}^P \cdot \mathbf{a}_1 \\ \mathbf{x}^P \cdot \mathbf{a}_2 \\ \alpha^D \end{bmatrix} \tag{6.7}$$

Define the generalized speeds as the time derivatives of the generalized coordinates:

$$\begin{aligned} \mathbf{u}_{arm_1} &= \dot{\mathbf{q}}_{arm_1} \\ \mathbf{u}_{arm_2} &= \dot{\mathbf{q}}_{arm_2} \\ \mathbf{u}_{obj} &= \dot{\mathbf{q}}_{obj} \end{aligned} \tag{6.8}$$

Rather than generating the reduced set of equations of motion, *system concatenation* retains the full seven degrees of freedom and additionally models the constraints, resulting in the following equations

of motion:

$$\begin{bmatrix} \mathbf{F}_{arm_1} \\ \mathbf{F}_{arm_2} \\ \mathbf{F}_{obj} \end{bmatrix} + \mathbf{F}_C = \begin{bmatrix} \mathbf{M}'_{arm_1} & \mathbf{0} & \mathbf{0} \\ & \mathbf{M}'_{arm_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M}'_{obj} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_{arm_1} \\ \dot{\mathbf{u}}_{arm_2} \\ \dot{\mathbf{u}}_{obj} \end{bmatrix} \tag{6.9}$$

$$+ \begin{bmatrix} \mathbf{C}'_{arm_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}'_{arm_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}'_{obj} \end{bmatrix} \begin{bmatrix} \mathbf{u}_{arm_1} \\ \mathbf{u}_{arm_2} \\ \mathbf{u}_{obj} \end{bmatrix} \tag{6.10}$$

$$= \begin{bmatrix} \mathbf{Y}'_{arm_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}'_{arm_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Y}'_{obj} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}_{arm_1} \\ \boldsymbol{\theta}_{arm_2} \\ \boldsymbol{\theta}_{obj} \end{bmatrix} \tag{6.11}$$

and motion constraints:

$$\mathbf{0}_{4\times 1} = \begin{bmatrix} \dot{\mathbf{x}}^{Q_1} - \dot{\mathbf{x}}^{Q'_1} \\ \dot{\mathbf{x}}^{Q_2} - \dot{\mathbf{x}}^{Q'_2} \end{bmatrix} \tag{6.12}$$

where $\mathbf{0}_{4\times 1}$ represents a 4×1 vector of zeroes.

6.2 Motion Constraints

The motion constraints in a system of multiple manipulators grasping an object are best represented by constraints on relative velocities—in both translation and orientation—between each manipulator end effector and its grasp point on the object. Specifically, the relative velocities are constrained to be zero. As Section 5.5 showed, these constraints can be adequately represented by *task*-space control objectives.

Figure 6.2 illustrates the constraints for multiple manipulators. The relative velocity between each end effector Q_i and its grasp point Q'_i satisfies the relationship:

$$\dot{\mathbf{x}}_{C_i} = \dot{\mathbf{x}}^{Q_i} - \dot{\mathbf{x}}^{Q'_i} \tag{6.13}$$

$$\underset{(5.22)}{=} \sum_{r=1}^{n} \mathbf{v}_r^{Q_i} u_r - \sum_{r=1}^{n} \mathbf{v}_r^{Q'_i} u_r \tag{6.14}$$

$$\underset{(5.24)}{=} \left[\mathbf{v}_r^{Q_i}\right] \mathbf{u} - \left[\mathbf{v}_r^{Q'_i}\right] \mathbf{u} \tag{6.15}$$

$$= \left(\left[\mathbf{v}_r^{Q_i}\right] - \left[\mathbf{v}_r^{Q'_i}\right]\right) \mathbf{u} \tag{6.16}$$

where $\dot{\mathbf{x}}_{C_i} \in \mathbb{R}^2$ is the relative velocity between the two points, and $\left[\mathbf{v}_r^{Q_i}\right] \in \mathbb{R}^{2\times 7}$ and $\left[\mathbf{v}_r^{Q'_i}\right] \in \mathbb{R}^{2\times 7}$ are the matrix of partial velocity vectors associated with Q_i and Q'_i, respectively.

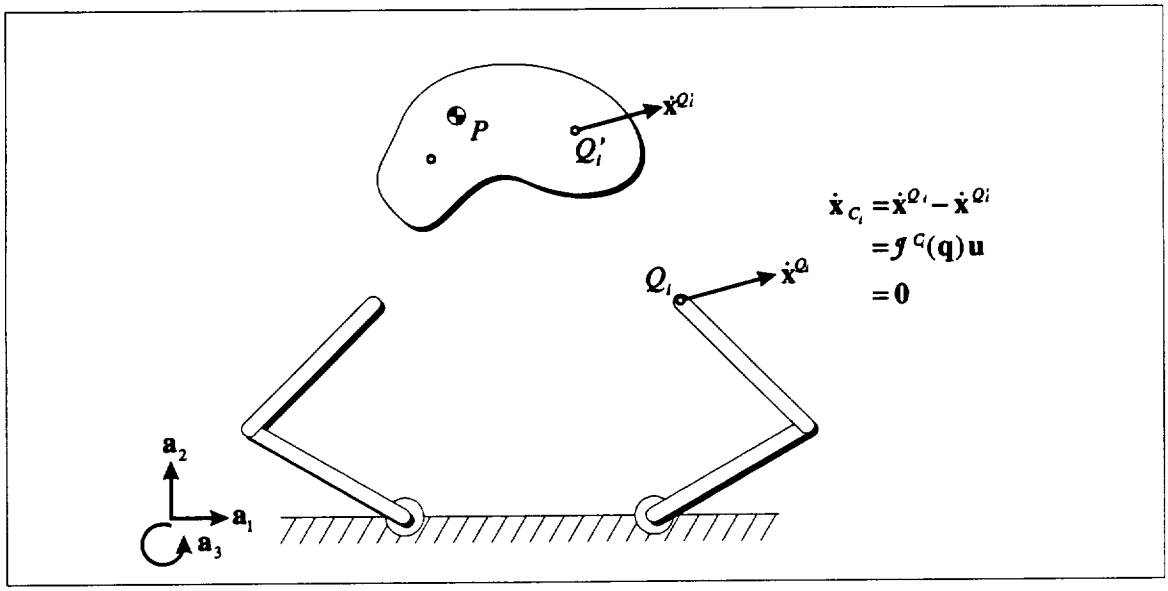


Figure 6.2: **Motion Constraints**

A motion constraint exists at each grasp point. Q_i is a endpoint of manipulator i, and Q_i' is the associated grip port on the payload object. $\dot{\mathbf{x}}^{Q_i}$ is the endpoint velocity, and $\dot{\mathbf{x}}^{Q_i'}$ is the velocity of the grip port. When the object is grasped, the relative velocity between Q_i and Q_i' is constrained to be zero. The constraint velocity $\dot{\mathbf{x}}_{C_i}$ can be expressed in terms of a generalized Jacobian $\mathcal{J}^{C_i}$ and the generalized speeds $\mathbf{u}$. $\mathbf{a}_1$, $\mathbf{a}_2$, *and* $\mathbf{a}_3$ *are mutually orthogonal unit vectors fixed in the inertial frame.*

Defining a generalized Jacobian component for the ith relative velocity $\dot{\mathbf{x}}_{C_i}$ as

$$\mathcal{J}^{C_i}(\mathbf{q}) \triangleq \left[\mathbf{v}_r^{Q_i}\right] - \left[\mathbf{v}_r^{Q_i'}\right] \tag{6.17}$$

the relative velocity can thus be expressed as:

$$\dot{\mathbf{x}}_{C_i} \underset{(6.16,6.17)}{=} \mathcal{J}^{C_i}(\mathbf{q})\mathbf{u} \tag{6.18}$$

In general, there also may be rotational constraints at the end effector. If the manipulator has a solid grasp of the object, for example, then the relative angular velocity between the end effector end_j and the payload D must be zero. These can be represented by:

$$\omega_{C_j} = \omega^{end_j} - \omega^D \tag{6.19}$$

$$\underset{(5.21)}{=} \sum_{r=1}^{n} \omega_r^{end_j} u_r - \sum_{r=1}^{n} \omega_r^D u_r \tag{6.20}$$

$$\underset{(5.23)}{=} \left[\omega_r^{end_j}\right]\mathbf{u} - \left[\omega_r^D\right]\mathbf{u} \tag{6.21}$$

$$= \left(\left[\omega_r^{end_j}\right] - \left[\omega_r^D\right]\right)\mathbf{u} \tag{6.22}$$

$$= \mathcal{J}_\omega^{C_j}(\mathbf{q})\mathbf{u} \tag{6.23}$$

The expression for angular velocity constraints is completely analogous to that of the linear velocity constraints of Equation (6.18). Hence, angular velocity constraints also can be incorporated into the *task*-space control vector.

When the payload is grasped, these relative velocities are constrained to be zero. To enforce these constraints, the *task*-space controller incorporates these equations into the *task*-space control vector along with the payload object motions:

$$\dot{\mathbf{y}}^{task} = \begin{bmatrix} \dot{\mathbf{x}}_{obj} \\ \dot{\mathbf{x}}_{C_1} \\ \vdots \\ \dot{\mathbf{x}}_{C_N} \\ \omega_{C_1} \\ \vdots \\ \omega_{C_N} \end{bmatrix} \tag{6.24}$$

The only requirement is that $\mathbf{y}^{task}$ has the same degrees of freedom as the open-chain system.

By specifying zero for the constraint terms in the *desired task*-space vectors—$\mathbf{y}_d^{task}$, $\dot{\mathbf{y}}_d^{task}$, and $\ddot{\mathbf{y}}_d^{task}$—the trajectory generator requests motions in the system that are completely consistent with the motion constraints. The controller simultaneously enforces these constraints and the desired payload motions.

Example

Using the same example in Figure 6.2 with a pair of planar arm and a rigid payload object, choose the *task*-space control velocity vector to be a combination of the payload Cartesian and angular velocities and the relative velocities between the end effectors and their grasp points[5]:

$$\dot{\mathbf{y}}^{task} = \begin{bmatrix} \mathbf{u}_{obj} \\ \dot{\mathbf{x}}_{C_1} \\ \dot{\mathbf{x}}_{C_2} \end{bmatrix} \tag{6.25}$$

[5]Assume that the end effectors have pin connections, so that there are no rotational motion constraints at the end effectors.

where $\mathbf{u}_{obj}$ are the generalized speeds associated with the object, and $\dot{\mathbf{x}}_{C_1}$ and $\dot{\mathbf{x}}_{C_2}$ are the motion constraints for the two manipulator end effectors. The associated generalized Jacobian is:

$$\mathcal{J}(\mathbf{q}) = \left[\begin{array}{c|ccc} & 1 & 0 & 0 \\ \mathbf{0}_{4\times 4} & 0 & 1 & 0 \\ & 0 & 0 & 1 \\ \hline \multicolumn{4}{c}{\mathcal{J}^{C_1}(\mathbf{q})} \\ \multicolumn{4}{c}{\mathcal{J}^{C_2}(\mathbf{q})} \end{array}\right] \tag{6.26}$$

which utilizes the constraint Jacobians of Equation (6.17).

Since the relative velocities between each end effector and its grasp point is constrained to be zero, the controller simply sets the last four elements of the *desired task*-space position, velocity, and acceleration vectors—$\mathbf{y}_d^{task}$, $\dot{\mathbf{y}}_d^{task}$, and $\ddot{\mathbf{y}}_d^{task}$—to zero.

6.3 Force Constraints

In addition to motion constraints, there are constraint forces at the manipulator end effectors. The end effectors must exert forces on the object to make it move. This section determines the nature of those constraint forces.

Figure 6.3 shows the two resultant forces at each grasp point. The endpoint exerts a force $\mathbf{f}^{Q_i'}$ on the object at Q_i', and the object exerts a force $\mathbf{f}^{Q_i}$ on the endpoint Q_i. These forces are equal but pointing in opposite directions:

$$\mathbf{f}^{Q_i'} = -\mathbf{f}^{Q_i} \tag{6.27}$$

Analogous relationships exists for constraint moments $\mathbf{m}$. These constraint forces and moments can be expressed as generalized constraint forces:

$$\mathbf{F}_C \underset{(5.34)}{=} \sum_{\substack{\text{all} \\ \text{constraint} \\ \text{forces } k}} \left[\mathbf{v}_r^k\right]^T \mathbf{f}^k + \sum_{\substack{\text{all} \\ \text{constraint} \\ \text{moments } l}} \left[\boldsymbol{\omega}_r^l\right]^T \mathbf{m}^l \tag{6.28}$$

Combining the two forces at each grasp point halves the number of terms in its sum, and combining the two moments at each grasp point points also halves the number of terms in its sum. These are

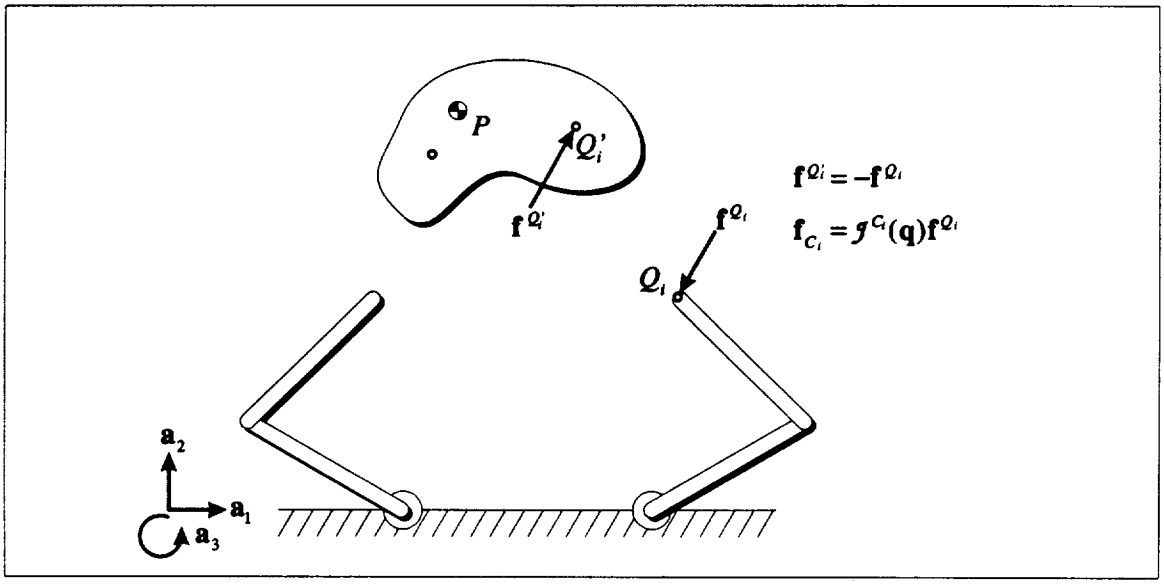


Figure 6.3: **Force Constraints**

Q_i is a point coincident with the ith end effector, and Q_i' is coincident with the ith grasp point on the object. $\mathbf{f}^{Q_i}$ is the force on the end effector passing through Q_i, and $\mathbf{f}^{Q_i'}$ is the force on the object passing through Q_i'. The constraint force $\mathbf{F}_{C_i}$ can be expressed in terms of the same generalized Jacobian $\mathcal{J}^{C_i}$ utilized for the constraint velocities.

indicated by the changes in the summing indices from k to i and l to j:

$$
\begin{aligned}
\mathbf{F}_C &= \sum_{\substack{\text{all}\\ \text{constrained}\\ \text{grasps } i}} \left[\mathbf{v}_r^{Q_i}\right]^T \mathbf{f}^{Q_i} + \left[\mathbf{v}_r^{Q_i'}\right]^T \mathbf{f}^{Q_i'} \\
&\quad + \sum_{\substack{\text{all}\\ \text{constrained}\\ \text{grasps } j}} \left[\boldsymbol{\omega}_r^{end_j}\right]^T \mathbf{m}^{end_j} + \left[\boldsymbol{\omega}_r^{D}\right]^T \mathbf{m}^{grasp_j'} \\
&\underset{(6.27)}{=} \sum_{\substack{\text{all}\\ \text{constrained}\\ \text{grasps } i}} \left[\mathbf{v}_r^{Q_i}\right]^T \mathbf{f}^{Q_i} - \left[\mathbf{v}_r^{Q_i'}\right]^T \mathbf{f}^{Q_i} \\
&\quad + \sum_{\substack{\text{all}\\ \text{constrained}\\ \text{grasps } j}} \left[\boldsymbol{\omega}_r^{end_j}\right]^T \mathbf{m}^{end_j} - \left[\boldsymbol{\omega}_r^{D}\right]^T \mathbf{m}^{end_j} \\
&= \sum_{\substack{\text{all}\\ \text{constrained}\\ \text{grasps } i}} \left(\left[\mathbf{v}_r^{Q_i}\right]^T - \left[\mathbf{v}_r^{Q_i'}\right]^T\right) \mathbf{f}^{Q_i}
\end{aligned}
$$

$$+ \sum_{\substack{\text{all} \\ \text{constrained} \\ \text{grasps } j}} \left(\left[\omega_r^{end_j} \right]^T - \left[\mathbf{v}_r^D \right]^T \right) \mathbf{m}^{end_j}$$

$$\underset{(6.17)}{=} \sum_{\substack{\text{all} \\ \text{constrained} \\ \text{grasps } i}} \mathcal{J}^{C_i T}(\mathbf{q}) \mathbf{f}^{Q_i} + \sum_{\substack{\text{all} \\ \text{constrained} \\ \text{grasps } j}} \mathcal{J}_\omega^{C_j T}(\mathbf{q}) \mathbf{m}^{end_j} \tag{6.29}$$

The generalized Jacobian components, $\mathcal{J}^{C_i}(\mathbf{q})$ and $\mathcal{J}_\omega^{C_j}(\mathbf{q})$, in Equation (6.29) are the *same* ones that appear in the velocity and angular velocity constraints (see Equation (6.18) and Equation (6.23)).

Hence, for a system with N end effectors, the generalized constraint force can be written in matrix notation as:

$$\mathbf{F}_C \underset{(6.29)}{=} \begin{bmatrix} \mathcal{J}^{C_1}(\mathbf{q}) & \cdots & \mathcal{J}^{C_p}(\mathbf{q}) & \mathcal{J}_\omega^{C_1}(\mathbf{q}) & \cdots & \mathcal{J}_\omega^{C_s}(\mathbf{q}) \end{bmatrix}^T \begin{bmatrix} \mathbf{f}^{Q_1} \\ \vdots \\ \mathbf{f}^{Q_N} \\ \mathbf{m}^{end_1} \\ \vdots \\ \mathbf{m}^{end_N} \end{bmatrix} \tag{6.30}$$

Define the *constraint Jacobian* $\mathcal{J}^C(\mathbf{q})$ as the first matrix of Equation (6.30):

$$\mathcal{J}^C(\mathbf{q}) \triangleq \begin{bmatrix} \mathcal{J}^{C_1}(\mathbf{q}) & \cdots & \mathcal{J}^{C_p}(\mathbf{q}) & \mathcal{J}_\omega^{C_1}(\mathbf{q}) & \cdots & \mathcal{J}_\omega^{C_s}(\mathbf{q}) \end{bmatrix} \tag{6.31}$$

Those terms corresponding to end-effector degrees of freedom that are not constrained may be dropped.

Example

For the example in Figure 6.3, the generalized constraint force vector is given by:

$$\mathbf{F}_C = \begin{bmatrix} \mathcal{J}^{C_1}(\mathbf{q}) & \mathcal{J}^{C_2}(\mathbf{q}) \end{bmatrix}^T \begin{bmatrix} \mathbf{f}^{Q_1} \\ \mathbf{f}^{Q_2} \end{bmatrix} \tag{6.32}$$

where $\mathcal{J}^{C_1}(\mathbf{q})$ and $\mathcal{J}^{C_2}(\mathbf{q})$ are the constraint Jacobians for the two end effectors, and $\mathbf{f}^{Q_1}$ and $\mathbf{f}^{Q_2}$ are the end-effector forces.

6.4 Force and Torque Mapping

The *task*-space adaptive controller of Equation (5.8), however, cannot be implemented directly with multiple-manipulators utilizing *system concatenation*. One small modification remains. Because *system*

concatenation model represents the payload object as a separate system, the commanded generalized active forces resulting from the control law contain elements that correspond to forces and moments on the object. Since the object does not have actuators of its own, these forces and moments must be supplied by the manipulators. An additional mapping is required to map the commanded forces and moments on the object to commanded manipulator forces and moments.

Consider the two planar arms and object in Figure 6.1 as an example. The generalized active force for the system is (see Equation (6.11)):

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{arm_1} \\ \mathbf{F}_{arm_2} \\ \mathbf{F}_{obj} \end{bmatrix} \tag{6.33}$$

The generalized forces on the object, represented by $\mathbf{F}_{obj}$, must be mapped back to the generalized active forces corresponding to the manipulators, $\mathbf{F}_{arm_1}$ and $\mathbf{F}_{arm_2}$.

This section develops this force and moment mapping by explicitly modelling the constraint forces and moments at the manipulator end effectors. This approach first solves for the required end-effector forces and moments required to generate the requested force and torque on the object resulting from the control law, it then uses the result from Section 6.3 to transform the end-effector forces and moments into generalized constraint forces. The sum of the generalized active forces and the generalized constraint forces forms a pseudo-generalized active force vector $\mathcal{F}$ such that the *system concatenated* model in Equation (6.3) can be used directly in the *task*-space adaptive control law, Equation (5.8), with $\mathbf{F}$ replaced by $\mathcal{F}$.

6.4.1 End-Effector Constraint Forces and Moments

The required end-effector forces and moments needed to apply the desired force and torque on the object is not unique, as the free-body diagram in Figure 6.4 illustrates. The vector $\mathbf{R} \in \mathbb{R}^3$ represents the resultant of all the end-effector forces and acts on a line passing through the center of mass of the object[6], and $\mathbf{T} \in \mathbb{R}^3$ is the total torque placed on the object by the end-effector forces and moments. At each grasp point Q_i', there may be a force $\mathbf{f}_i \in \mathbb{R}^3$ and a moment $\mathbf{m}_i \in \mathbb{R}^3$, and each $\mathbf{p}_i \in \mathbb{R}^3$ represents the position vector from the center of mass to the associated grasp point Q_i'.

[6]If the *task*-space control objective is to control the position of some other point P on the object, then the analysis uses the point P rather than the center of mass. P is commonly known as the *remote center of compliance*[32, 49].

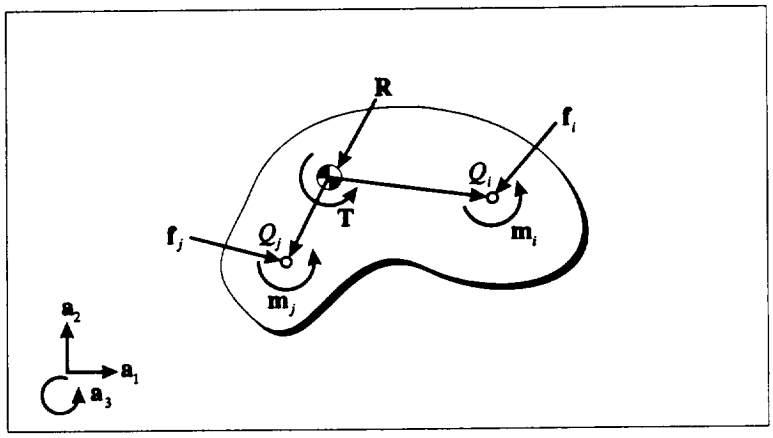


Figure 6.4: **Object Free-Body Diagram**

$\mathbf{R}$ is the resultant of all forces acting on the object whose line of action passes through the center of mass of the object, $\mathbf{T}$ is the total torque on the object, $\mathbf{f}_i$ and $\mathbf{m}_i$ are the forces and moments applied at point Q_i', and $\mathbf{p}_i$ is the position vector from the center of mass to point Q_i'. The vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ are mutually orthogonal unit vectors fixed in the inertial frame.

For a manipulator system with N end effectors, the resultant force $\mathbf{R}$ and the torque $\mathbf{T}$ can be related to the end-effector forces and moments as:

$$\begin{bmatrix} \mathbf{R} \\ \mathbf{T} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} \mathbf{f}_i \\ \sum_{j=1}^{N} \mathbf{m}_j + \sum_{i=1}^{N} \mathbf{p}_i \times \mathbf{f}_i \end{bmatrix} \tag{6.34}$$

$$= \begin{bmatrix} \mathbf{I}_{3\times3} & \dots & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \dots & \mathbf{0}_{3\times3} \\ \mathbf{P}_1 & \dots & \mathbf{P}_N & \mathbf{I}_{3\times3} & \dots & \mathbf{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_N \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix} \tag{6.35}$$

where $\mathbf{P}_i \in \mathbb{R}^{3\times3}$ is the cross product matrix,

$$\mathbf{P}_i = \begin{bmatrix} 0 & -\mathbf{p}_i \cdot \mathbf{a}_3 & \mathbf{p}_i \cdot \mathbf{a}_2 \\ \mathbf{p}_i \cdot \mathbf{a}_3 & 0 & -\mathbf{p}_i \cdot \mathbf{a}_1 \\ -\mathbf{p}_i \cdot \mathbf{a}_2 & \mathbf{p}_i \cdot \mathbf{a}_1 & 0 \end{bmatrix} \tag{6.36}$$

Equation (6.35) represents an under-constrained set of equations, such that there is no unique inverse mapping from the resultant force and torque to end-effector forces and moments.

Let the $\mathbf{W}_{obj} \in \mathbb{R}^{6\times 6N}$ denote the mapping matrix:

$$\mathbf{W}_{obj} \triangleq \begin{bmatrix} \mathbf{I}_{3\times 3} & \cdots & \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} & \cdots & \mathbf{0}_{3\times 3} \\ \mathbf{P}_1 & \cdots & \mathbf{P}_N & \mathbf{I}_{3\times 3} & \cdots & \mathbf{I}_{3\times 3} \end{bmatrix} \tag{6.37}$$

and let $\mathbf{W}^{\dagger}_{obj} \in \mathbb{R}^{6N\times 6}$, the weighted pseudo-inverse of $\mathbf{W}_{obj}$ be:

$$\mathbf{W}^{\dagger}_{obj} \triangleq \mathbf{A}^{-1}\mathbf{W}^{T}_{obj}\left(\mathbf{W}_{obj}\mathbf{A}^{-1}\mathbf{W}^{T}_{obj}\right)^{-1} \tag{6.38}$$

where $\mathbf{A} \in \mathbb{R}^{6N\times 6N}$ is *any* weighting matrix and represents the desired manipulator load distribution. When $\mathbf{A}$ is the identity matrix, the solution weights all manipulators equally. The general solution for end-effector forces and moments now can be expressed as:

$$\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_N \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix} = \mathbf{W}^{\dagger}_{obj}\begin{bmatrix} \mathbf{R} \\ \mathbf{T} \end{bmatrix} + \left(\mathbf{I}_{6N\times 6N} - \mathbf{W}^{\dagger}_{obj}\mathbf{W}_{obj}\right)\mathbf{f}_{aux} \tag{6.39}$$

where $\mathbf{f}_{aux} \in \mathbb{R}^{6N}$ is any optional auxiliary input. Those terms in Equation (6.39) corresponding end-effector degrees of freedom that are not constrained may be dropped.

The premultiplier of $\mathbf{f}_{aux}$ chooses the portion of $\mathbf{f}_{aux}$ in the null space of $\mathbf{W}_{obj}$; this term represents internal loading on the object that generates no motion. Usually, this entire term can be replaced by $\mathbf{f}_{int}$ that is chosen to represent the desired internal object "squeeze" forces.

Utilizing the generalized constraint force of Equations 6.30 and the constraint Jacobian of 6.31, Equation (6.39) can be expressed as:

$$\mathbf{F}_C = \mathcal{J}^{CT}(\mathbf{q})\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_N \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}$$

$$\underset{(6.39)}{=} \mathcal{J}^{C^T}(\mathbf{q}) \left(\mathbf{W}_{obj}^{\dagger} \begin{bmatrix} \mathbf{R} \\ \mathbf{T} \end{bmatrix} + \mathbf{f}_{int} \right) \tag{6.40}$$

Hence, given the desired generalized active forces for the payload computed by the *task*-space control law (Equation (5.8)), Equation (6.40) produces the equivalent set of generalized constraint forces. This relationship is used in the following chapter to round out the development of the complete *task*-space adaptive controller.

Chapter 7

Task-Space Adaptive Controller

Combining *system concatenation* and *task*-space control into the same framework results in a very general adaptive controller for rigid-link robotic systems that can contain any number of cooperating manipulators and payload objects, operating in any number of control modes. This chapter formally presents the control and adaptive laws of the *task*-space adaptive controller, and summarizes its properties.

7.1 Control Law

Utilizing *system concatenation*, partition the robot system model into M actuated subsystems denoted by the subscripts arm_i and m unactuated subsystems denoted by subscripts obj_i. The system model can be expressed as[1]:

$$\mathbf{F} + \mathbf{F}_C = \mathbf{M}'(\mathbf{q})\dot{\mathbf{u}} + \mathbf{C}'(\mathbf{q},\mathbf{u})\mathbf{u} + \mathbf{G}'(\mathbf{q}) \tag{7.1}$$

$$\mathbf{0} = \dot{\mathbf{x}}_C \tag{7.2}$$

[1]This control framework is equally applicable for simpler robotic systems. For a single-manipulator system, there would be only one partition and no constraint equations.

where $\dot{\mathbf{x}}_C \in \mathbb{R}^N$ are N motion constraints and the partitions are:

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_{arm_1} \\ \vdots \\ \mathbf{q}_{arm_M} \\ \mathbf{q}_{obj_1} \\ \vdots \\ \mathbf{q}_{obj_m} \end{bmatrix} \tag{7.3}$$

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_{arm_1} \\ \vdots \\ \mathbf{u}_{arm_M} \\ \mathbf{u}_{obj_1} \\ \vdots \\ \mathbf{u}_{obj_m} \end{bmatrix} \tag{7.4}$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{arm_1} \\ \vdots \\ \mathbf{F}_{arm_M} \\ \mathbf{F}_{obj_1} \\ \vdots \\ \mathbf{F}_{obj_m} \end{bmatrix} \tag{7.5}$$

$$\mathbf{M}'(\mathbf{q}) = \begin{bmatrix} \mathbf{M}'_{arm_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M}'_{arm_M} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{M}'_{obj_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{M}'_{obj_m} \end{bmatrix} \tag{7.6}$$

$$\mathbf{C}'(\mathbf{q},\mathbf{u}) = \begin{bmatrix} \mathbf{C}'_{arm_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}'_{arm_M} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{C}'_{obj_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{C}'_{obj_m} \end{bmatrix} \tag{7.7}$$

$$\mathbf{G}'(\mathbf{q}) = \begin{bmatrix} \mathbf{G}'_{arm_1} \\ \vdots \\ \mathbf{G}'_{arm_M} \\ \mathbf{G}'_{obj_1} \\ \vdots \\ \mathbf{G}'_{obj_m} \end{bmatrix} \tag{7.8}$$

Additionally define the pseudo-generalized active force $\mathcal{F}$ to be:

$$\mathcal{F} \triangleq \mathbf{F} + \mathbf{F}_C \tag{7.9}$$

The control law for the *task*-space adaptive controller now can be expressed as:

$$\mathcal{F} = \widehat{\mathbf{M}}'(\mathbf{q})\dot{\mathbf{u}}_d + \widehat{\mathbf{C}}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d + \widehat{\mathbf{G}}'(\mathbf{q}) + \mathbf{f}_t + \mathcal{J}^T(\mathbf{q})\left(\mathbf{K}_{Vy}\dot{\tilde{\mathbf{y}}}^{task} + \mathbf{K}_{Py}\tilde{\mathbf{y}}^{task}\right) \tag{7.10}$$

$$\tau = \mathbf{W}^T(\mathbf{q}) \begin{bmatrix} \mathbf{I}_{M\times M} & -\mathcal{J}^{C^T}(\mathbf{q})\mathbf{W}^{\dagger}_{obj} \\ \mathbf{0} & \mathbf{I}_{m\times m} \end{bmatrix} \left(\mathcal{F} - \mathcal{J}^{C^T}(\mathbf{q})\mathbf{f}_{int}\right) \tag{7.11}$$

where $\dot{\mathbf{y}}^{task}$ includes the motion constraints $\dot{\mathbf{x}}_C$ and the corresponding terms for the constraints in the desired *task*-space trajectories—$\mathbf{y}_d^{task}$, $\dot{\mathbf{y}}_d^{task}$, and $\ddot{\mathbf{y}}_d^{task}$—are set to zero. See Section 7.3 for the derivation of the actuator mapping of Equation (7.11). The control law can be expressed in terms of the parameter vector as:

$$\mathcal{F} \underset{(2.9)}{=} \mathbf{Y}'(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\hat{\boldsymbol{\theta}} + \mathbf{f}_t + \mathcal{J}^T(\mathbf{q})\left(\mathbf{K}_{Vy}\dot{\tilde{\mathbf{y}}}^{task} + \mathbf{K}_{Py}\tilde{\mathbf{y}}^{task}\right) \tag{7.12}$$

The adaptive update law is:

$$\dot{\hat{\boldsymbol{\theta}}} = \Gamma\mathbf{Y}'^T(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\mathcal{J}^{-1}(\mathbf{q})\left(\dot{\tilde{\mathbf{y}}}^{task} + c\tilde{\mathbf{y}}^{task}\right) \tag{7.13}$$

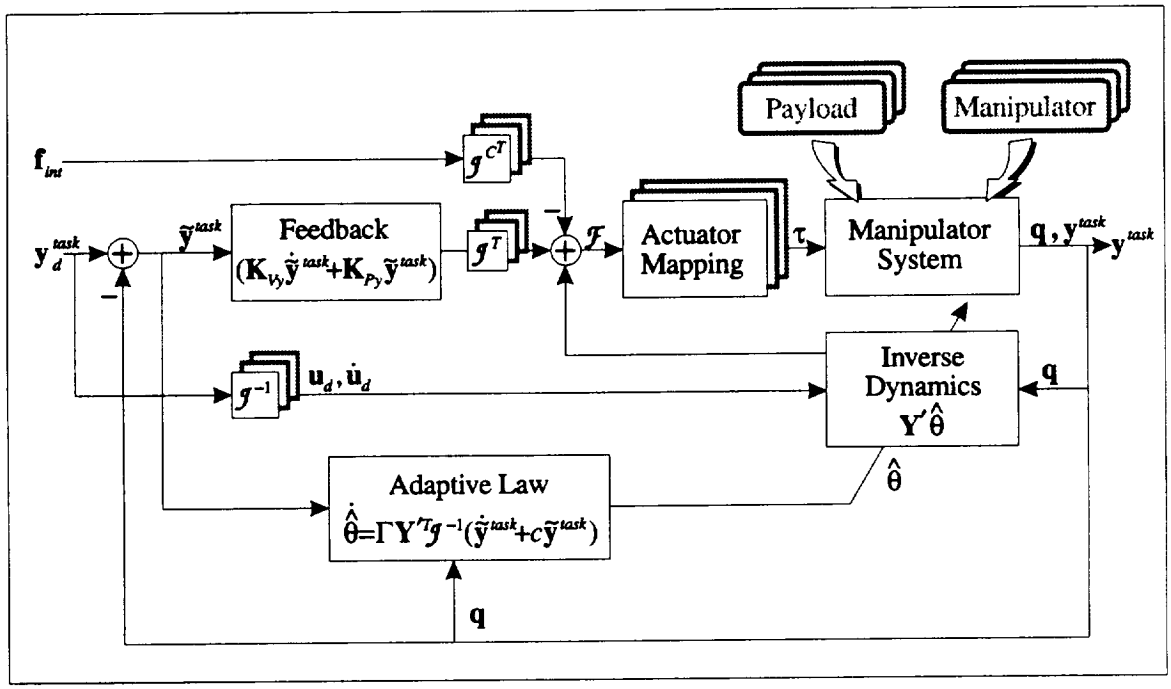


Figure 7.1: **Block Diagram of the *Task*-Space Adaptive Controller**

The task-space adaptive control tracks task-space trajectories by performing feedback directly on the task-space errors, $\tilde{\mathbf{y}}^{task}$. The adaptive parameter update is also based on the task-space tracking error. This controller structure differs from the single-manipulator task-space adaptive controller, Figure 5.1, by the more sophisticated actuator mapping, given by Equation (7.11). The desired internal forces on the payload, $\mathbf{f}_{int}$, also may be specified. Generalized Jacobians transform the task-space values into equivalent generalized speeds values.

Figure 7.1 shows the block diagram of the complete *task*-space adaptive controller, incorporating multiple-manipulator control. The grayed manipulator system blocks indicate that the *task*-space adaptive controller is capable of handling any number of manipulators and objects, and they may be added to the system at any time, as long as the associated Jacobians are updated accordingly. The cooperative manipulator also requires a more sophisticated force- and moment-mapping block given by Equation (7.11) to derive the actuator forces and torques from the control law. This actuator mapping reduces to the identity matrix when the robot is not performing payload object manipulation, so this controller structure is a superset of the single-manipulator *task*-space adaptive controller of Figure 5.1. The internal "squeeze" force, $\mathbf{f}_{int}$, on the payload object may be specified, a most valuable capability when handling fragile objects.

The block diagram also indicates that switching control modes is a simple matter of switching to the

appropriate Jacobian and the appropriate force- and torque-mapping blocks. The valid control modes can range from single manipulator joint control to cooperative multiple-manipulator and multiple-payload object control from the free-flying base of a space robot.

7.2 Controller Properties

The following items summarize the features of the *task*-space adaptive controller.

- *Task-space control.* The *task*-space is an abstraction of control modes. The controller changes control modes by changing the definition of the *task*-space control vector. Thus, the *task*-space controller is capable of handling any number or mixture of control modes from cooperative multiple-manipulator control on a free-flying base to single-arm joint control. The feedback is performed directly in the chosen *task*-space to minimize *task*-space trajectory tracking errors.

- *Inverse-dynamics controller.* The *task*-space adaptive controller is a model-based controller. The "Inverse Dynamics" block compensates for the inertial effects and the nonlinear Coriolis and centrifugal force terms. The adaptive controller updates this inverse dynamics model to minimize trajectory tracking errors. For closed-kinematic-chain systems, *system concatenation* is utilized to simplify the inverse-dynamics model.

- *Tracking-error adaptive control.* The parameter vector $\widehat{\theta}$ used in the inverse-dynamics block of the controller is adaptively updated. The parameters, and thus the plant model, are updated to minimize *task*-space trajectory tracking errors.

- *Separable adaptation parameters.* For complex systems with multiple subsystems, *system concatenation* modelling keeps separate the adaptable parameters of each subsystem. Doing so allows separate adaptation of each subsystem without sacrificing the capability for simultaneous adaptation of the entire system.

- *Generalized Jacobian.* The generalized Jacobian is the key to *task*-space control. Any quantity that satisfies the generalized Jacobian relationship (see Equation (5.7)) can be utilized as a *task*-space control vector. Changing control modes requires changing the generalized Jacobian blocks to match the new *task*-space control vector.

- *Actuator mapping.* For implementation, an additional mapping is required to transform generalized active forces—or pseudo-generalized active forces for closed-kinematic-chain systems—into actuator forces and torques.

- *Controller insensitivity to sensor noise.* As with the original joint-space adaptive controller, the new *task*-space controller is insensitive to sensor noise during regulation. Because the the inverse-dynamics feedforward block utilizes desired, not measured, trajectory velocities and accelerations, sensor noise does not produce unwanted feedforward actuation.

- *Adaptation insensitivity to sensor noise.* Since the adaptation update law also utilizes the regressor, $\mathbf{Y}(\dot{\mathbf{q}}, \mathbf{u}_d, \mathbf{u}_d, \dot{\mathbf{u}}_d)$, that contains desired velocities and accelerations, it is not sensitive to sensor noise during regulation. The parameters do not drift.

7.3 Actuator Mapping

The actuator mapping utilizes the generalized-constraint-force expression of Equation (6.40) to replace $\mathbf{F}_C$ in the expression for the pseudo-generalized active force $\mathcal{F}$:

$$\begin{aligned}
\mathcal{F} &\underset{(6.30,6.39)}{=} \begin{bmatrix} \mathbf{F}_{arm_1} \\ \vdots \\ \mathbf{F}_{arm_M} \\ \mathbf{F}_{obj_1} \\ \vdots \\ \mathbf{F}_{obj_m} \end{bmatrix} + \mathcal{J}^{C^T}(\mathbf{q}) \left(\mathbf{W}_{obj}^{\dagger} \begin{bmatrix} \mathbf{F}_{obj_1} \\ \vdots \\ \mathbf{F}_{obj_m} \end{bmatrix} + \mathbf{f}_{int} \right) \\
&= \begin{bmatrix} \mathbf{I}_{M\times M} & \mathcal{J}^{C^T}(\mathbf{q})\mathbf{W}_{obj}^{\dagger} \\ \mathbf{0} & \mathbf{I}_{m\times m} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{arm_1} \\ \vdots \\ \mathbf{F}_{arm_M} \\ \mathbf{F}_{obj_1} \\ \vdots \\ \mathbf{F}_{obj_m} \end{bmatrix} + \mathcal{J}^{C^T}(\mathbf{q})\mathbf{f}_{int} \\
&= \begin{bmatrix} \mathbf{I}_{M\times M} & \mathcal{J}^{C^T}(\mathbf{q})\mathbf{W}_{obj}^{\dagger} \\ \mathbf{0} & \mathbf{I}_{m\times m} \end{bmatrix} \mathbf{F} + \mathcal{J}^{C^T}(\mathbf{q})\mathbf{f}_{int} \qquad (7.14)
\end{aligned}$$

where $\mathcal{J}^C(q)$ is the matrix of constraint Jacobians defined in Section 6.2. The inverse relationship needed in the control law is[2]:

$$\begin{aligned} \tau &= \mathbf{W}^T(\mathbf{q})\mathbf{F} \\ &= \mathbf{W}^T(\mathbf{q}) \begin{bmatrix} \mathbf{I}_{M\times M} & -\mathcal{J}^{C^T}(\mathbf{q})\mathbf{W}_{obj}^{\dagger} \\ \mathbf{0} & \mathbf{I}_{m\times m} \end{bmatrix} \left(\mathcal{F} - \mathcal{J}^{C^T}(\mathbf{q})\mathbf{f}_{int}\right) \end{aligned} \tag{7.15}$$

7.4 Conclusions

The *system concatenation* concept furnishes to the *task*-space adaptive controller the ability to control multiple cooperating manipulators. Moreover, it does so in an efficient manner by not requiring the complex closed-kinematic-chain equations of motion. Cooperative manipulator control incorporates motion constraints at the end effectors into the *task*-space control vector, and utilizes the explicit modelling of end-effector constraint forces and moments to generate a mapping from generalized active forces and generalized constraint forces to actuator forces and torques.

The resulting control law equations are of higher order than those resulting from the use of the closed-kinematic-chain equations of motion, but the simplicity of the block-diagonal nature of the *concatenated* system equations more than offsets the disadvantages of the increased order. Additionally, the generalized Jacobians tend to be sparse matrices, and sophisticated matrix computation algorithms can take advantage of that to reduce further the computational burden. Finally, the use of *system concatenation* does not prevent the *task*-space controller structure from handling simpler single-arm control: All that is needed is the appropriate generalized Jacobian.

[2]The following matrix relationship is true for any matrix $\mathbf{W} \in \mathbb{R}^{n\times m}$:

$$\begin{bmatrix} \mathbf{I}_{n\times n} & \mathbf{W} \\ \mathbf{0} & \mathbf{I}_{m\times m} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{I}_{n\times n} & -\mathbf{W} \\ \mathbf{0} & \mathbf{I}_{m\times m} \end{bmatrix}$$

Chapter 8

The Experimental System

This chapter briefly describes the experimental system. More details can be found in Ullman's thesis[43].

Adaptive control of a free-flying space robot motivated the development of the new *task*-space adaptive control framework. Therefore, the *task*-space adaptive controller has been implemented on the Multi-Manipulator Free-Flying Space Robot experiment in the Aerospace Robotics Laboratory (ARL) for verification. Utilizing air-bearing technology, this facility has pioneered the research in space robotics by simulating the drag-free conditions of space with high fidelity in the laboratory[43, 19, 1].

The Multi-Manipulator Free-Flying Space Robot was designed to meet some very specific goals:

- The robot must be completely self-contained and free-flying, containing all the key components to be found on a real space robot.
- The space robot must be capable of autonomous operations.
- User interaction with the robot must occur at a high level to ease the burden on the human operator.

These goals resulted in a relatively sophisticated experimental system, of which adaptive control is only a small part.

Because of the system complexity, modularity in both hardware and software design became a necessity.

The Multi-Manipulator Free-Flying Space Robot is self-contained, containing its own pressurized gas system for floatation and propulsion. It incorporates a pair of arms for performing cooperative manipulation. The robot contains NiCad rechargeable batteries for use in free-flying experiments. Power

electronics can distribute electrical power utilizing either the batteries or external power. Additional analog electronics handle sensing and actuation, as well as on-board battery recharging. The robot also has a camera for on-board vision sensing. The on-board computer system provides the "brains" for the space robot and is the core of the controls system; the computers also provide on-board vision processing. Analog-to-Digital (A/D), and Digital-to-Analog (D/A), and Digital Input/Output (DIO) boards furnish the interface between the computers and the sensors and actuators. A wireless Ethernet module allows communication with the user and off-board computers[1].

Software modularity and the need to provide a high-level user-interface resulted in a hierarchical control approach. At the top level is a graphical user interface that allows the operator to view the robot and payloads and to direct robot actions via simple mouse motions and simple commands such as "move", "capture", or "release." At the heart of the control system is a *strategic controller* that processes the user input and schedules the necessary control mode changes based upon user and sensor inputs. The adaptive controller sits at a lower level, along with the trajectory generators. These too are modularized such that different controllers, trajectory generators, Jacobian calculations, etc., may be swapped in dynamically at the request of the strategic controller or the user. At the lowest level are the sensor and actuator nonlinear-compensation calculations.

8.1 Hardware Architecture

The Multi-Manipulator Free-Flying Space Robot is a completely self-contained and modularly designed robot capable of fully autonomous operation. The space robot floats on a cushion of air atop a 9 foot by 12 foot granite surface plate that is flat to within 0.001 inch between any two points on its surface. This results in extremely low friction, providing an accurate representation of the drag-free conditions of space in two dimensions. The robot operates in a plane perpendicular to the gravitational force; thus gravity is not a factor.

The space robot carries on board a pair of manipulators, a pressurize gas system for propulsion, batteries and electrical power systems, a complete set of sensors include a camera for vision sensing, a computer system, and a high-speed wireless communications system. Figure 8.1 shows a picture of the space robot[2]. The space robot is functionally separated into three layers. At the bottom is the

[1]Although high-speed communication is not required for user commands during operation, it greatly enhances controller development capabilities.

[2]The space robot was designed mainly by Marc Ullman and Ross Koningstein of the ARL. The vision system hardware—the

pressurized gas system. The spherical tanks, capable of holding 3000psi of compressed air, provides the gas supply for both flotation and propulsion. The middle layer contains the batteries, battery-charging system, power electronics, and analog electronics for the sensors. The top layer contains the computer system and digital electronics. It also holds the wireless Ethernet module. At the very top is the vision camera, sitting on its boom, that provides local sensing of the manipulator end effectors and of the payload objects. Finally, the pair of planar manipulators reaches out from the front of the robot.

The base of the space robot measures 500mm in diameter and, neglecting the camera boom, and 800mm in height. It has 65kg in mass and a moment of inertia about its center of mass of 3.2kg-m^2. The maximum reach of each arm is 600mm, and the mass of each arm is 2.3kg. Detailed mass distribution can be found in the initialization files in Appendix E.

8.1.1 Actuators

Free-flying base actuation is provided through eight gas-jet thrusters. The thrusters are mounted between the two bottom layers on the corners of a square whose diagonal is a diameter of the base (see Figure 8.2). The thrusters may be fired in combinations to produce translation and rotational motion. An optimal bang-off-bang thrust-mapping algorithm for the on-off thrusters is used to approximate linear control [43]. The pressure for the thrusters is regulated nominally at 100psi, and each thruster supplies 1N of force. Therefore, the maximum force that can be applied in any one direction is 2N, and the maximum torque is 1N-m.

The manipulators are in a SCARA configuration, each with two degrees of freedom. The actuators are brushless DC limited-angle torquers[3] mounted at the robot base to minimize link inertias and to prevent the center of mass of the robot from shifting away from the center of the air bearing. Each shoulder is connected directly to its motor, and each elbow is connected to its motor through a cable drive to eliminate backlash. Compensation for the nonlinear torque curve versus angular position is done in software[4]. The maximum torque from each shoulder motor is about 0.8N-m, and the maximum torque from each elbow motor is around 0.5N-m.

Each end effector is a pneumatically-actuated "gripper" that moves in the vertical direction on linear

Point Grabber II—is designed and built by the author.

[3]The motors are manufactured by Aeroflex. Model V40Y-6H is used for the shoulders and model V40Y-5H is used at the elbows.

[4]The torque curve is modelled as a fourth-order polynomial constrained to have only one inflection point at the motor angle with maximum torque output.

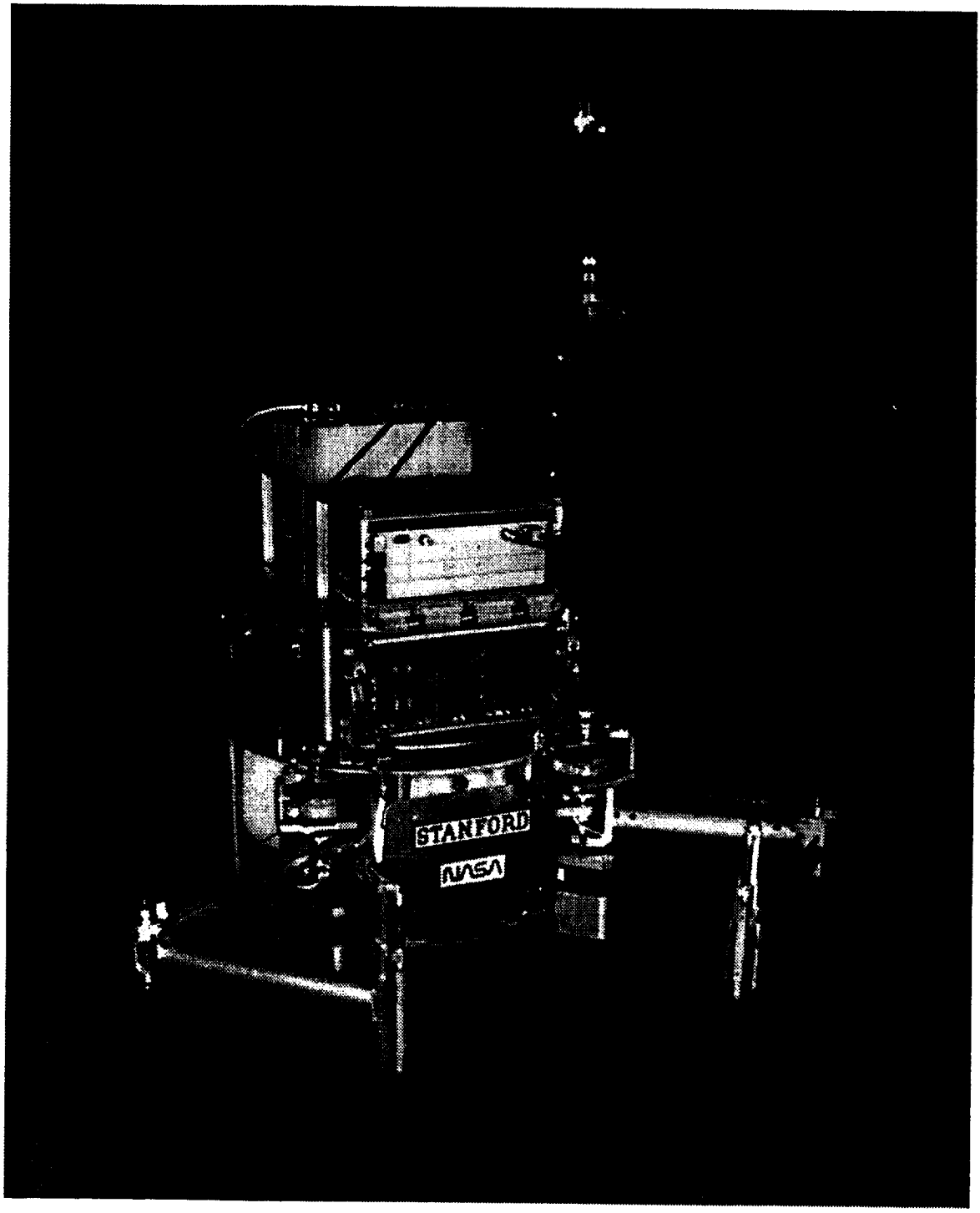


Figure 8.1: **Multi-Manipulator Free-Flying Space Robot**

The Multi-Manipulator Free-Flying Space Robot is a fully self-contained autonomous robot with on-board fuel, power, computers, vision, and wireless communications systems.

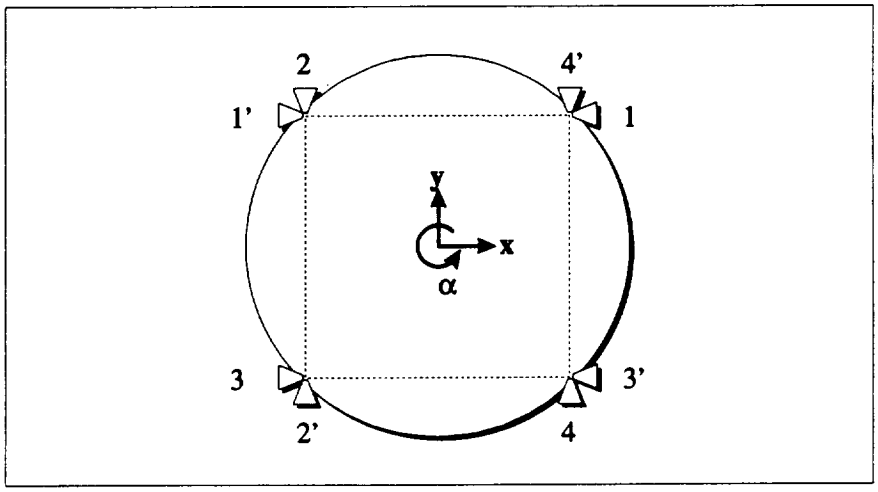


Figure 8.2: **Thruster Arrangement**

The space robot carries eight gas jet thrusters, mounted on the four corners of a square in the base of the robot. Two thrusters at each face of the square can provide pure translational motion, and when fired in combination, the thrusters can supply both rotational and translational motions simultaneously. The numbered pairs of thrusters provide positive and negative thrust along the indicated line of action. An optimal bang-off-bang thrust mapping is used to approximate linear control.

bearings. To grasp a payload object, the "grippers" are lowered into mating "gripper ports" in the payload. A bearing at the tip of the gripper prevents the gripper from applying a torque on the object, and an O-ring around the bearing ensures a snug fit in the gripper ports.

8.1.2 Sensors

The manipulator-joint-angle sensors are analog RVDT's (rotational variable differential transformers) mounted on the motor shafts. An additional analog filter provides band-limited pseudo-angular rate. The RVDT's are Pickering model 23501-0, each having a range of 150 degrees. The RVDT position signals are additionally passed through a third-order polynomial in software to compensate for nonlinearity.

Other sensors provide information on battery voltage and gas tank pressure. These are used during initialization to determine the health of the space robot system. They can also be monitored continuously during operation to determine when the robot should return to its dock for recharging and refueling.

8.1.3 Vision Subsystem

The position sensors for the space robot and payload objects are vision-based. A vision system consists of a Pulnix 440S CCD camera, a Point Grabber II vision-processing board[5], and a computer board to process the vision data and to run the VisionServer software[6]. The VisionServer utilizes bright targets to identify and track named objects. Three targets on each object are sufficient for identification and for providing position and orientation information in two dimensions[7]. Objects of interest are marked by bright targets using either infrared LED's or highly-reflective discs. The PointGrabber locates bright pixels in the camera's field of view and stores their positions. Postprocessing assembles the bright pixels into groups corresponding to the targets before passing the information to the VisionServer.

The combination of vision hardware and software can provide at 60Hz resolution of better than a $1/20^{th}$ of a pixel; for a field of view of two meters, this resolution translates to about 0.25mm. Third-order polynomials correct for the wide-angle-lens distortions to provide good accuracy over the entire field of view.

Two vision systems provide the global positioning information over the entire surface of the granite surface plate[8]. The cameras are mounted above the table, and two off-board computers perform the vision processing. The global position and velocity information is sent to the space robot via the wireless Ethernet link. The space robot base and each payload object are equipped with a triad of infrared LED's, mounted in a unique pattern. These patterns are registered with the VisionServers, allowing them to identify and track each body.

The space robot also contains an on-board vision system for local sensing. Because the field of view is smaller, this vision system provides much higher resolution. The local vision system also enables high-speed local feedback, and is not affected by possible communication delays and drop-outs from global positioning systems. Each manipulator end effector also carries an infrared LED, allowing the on-board vision system to track the endpoints for endpoint control[9].

---

[5] See Appendix B for more details on the Point Grabber II.

[6] The VisionServer software was developed by Stanley Schneider at ARL.

[7] Ongoing research at ARL is investigating the use of five targets per object for acquiring the position and orientation in full three dimensions.

[8] This "surrogate" global positioning system most likely will be replaced in space by some combination of inertial navigation system and GPS receivers.

[9] Endpoint sensing is of course absolutely essential for capturing payloads, because the controller must match the positions and velocities of the end effectors with those of the gripper ports on the payload.

8.2 Electrical Subsystem

Electrical power is needed to operate the manipulators, the computers, and communications. A fully self-contained power subsystem, all the analog and digital electronics, and the computer systems for control are contained on board the space robot.

8.2.1 Analog Electronics

Analog circuitry is needed to supply electrical power to the robot system, and to process the signals from analog sensors.

On-board rechargeable NiCad batteries provide electrical power to the the space robot when it is free-flying. The batteries can supply ±12VDC at up to 15 Amps. Connecting external power when the space robot is docked also provides electrical power and simultaneously recharges the batteries. Off-board charging is available as well, and the battery packs can be replaced while the space robot is "live". The batteries may be engaged and disengaged via manual toggle switches or through computer control. Figure 8.3 shows a simple schematic of the analog electronics subsystem.

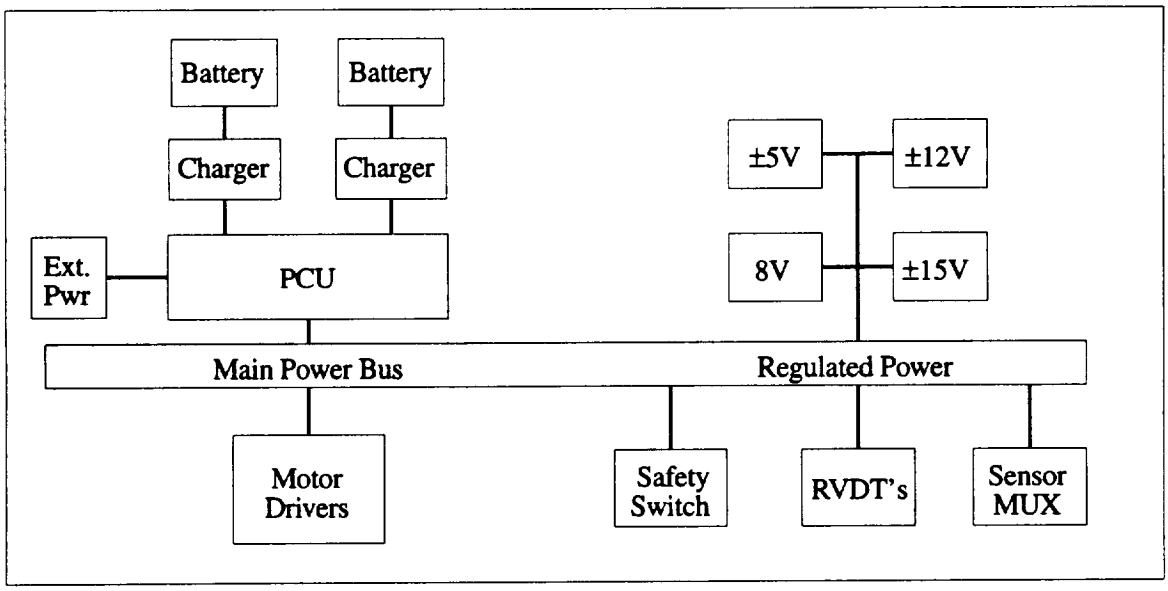


Figure 8.3: Analog Electronics Subsystem

The motor drivers use the raw, unregulated, power from the electrical bus. Power converter units

provide regulated power at ± 5V to the computer and digital electronics, ± 12V and ± 15V to the analog electronics subsystem, and $+8$V to the wireless Ethernet module.

The analog electronics subsystem contains the main power control unit (PCU), battery chargers, safety disconnect circuitry, sensor electronics, and motor drivers. The PCU contains the master power switch and the battery engaging switches[10]. It is the main power-distribution center, allowing any combination of batteries and external power to drive the power bus. The battery-charging circuits provide three charge rates and include automatic switching into trickle charge as the batteries reach their maximum voltage. The thrusters and arm motors are enabled through the safety disconnect switch. In addition to the manual enable switch, the controller must enable the switch explicitly during every sample period; otherwise, the safety switch kicks in to disconnect power to the motors and thruster relays. This effectively kills power to all actuators if the control system fails.

The sensor electronics contain the excitation circuitry and filtering for the RVDT's used for manipulator-joint-angle sensing. It also supplies power to the infrared LED's at the end effectors. A multiplexer board takes sensor readings from the battery voltages, gas tank pressure, and the regulated pressure. Finally, the motor drivers supply current to the arm motors.

8.2.2 Computer Subsystem

The real-time computer systems used in ARL are based on the VMEbus. The VMEbus is widely supported in industry with both hardware and software products. Using standardized products shortens development time and ensures robust, well-debugged computer components. Figure 8.4 show a simple schematic of the computer system.

The main on-board computer is a Motorola MV167 single board computer containing a 25MHz MC68040 processor. This computer handles all the *task*-space adaptive control calculations, high-level strategic control, trajectory calculations, path planning, sensor and actuator signal conditioning, and nonlinear compensation of sensor and actuator signals. Another computer, the Motorola MV133-1 with a 16.67MHz MC68020, performs all the calculations for the on-board vision system. Two additional off-board computers, Motorola MV147 20MHz MC68030 processor boards, handle the global vision system calculations. The on-board computer communicates with each other via the VMEbus backplane, and communication with the off-board computers utilizes the wireless Ethernet.

[10]The batteries may be engaged manually or through computer control. The switches automatically disengage when the batteries are removed.

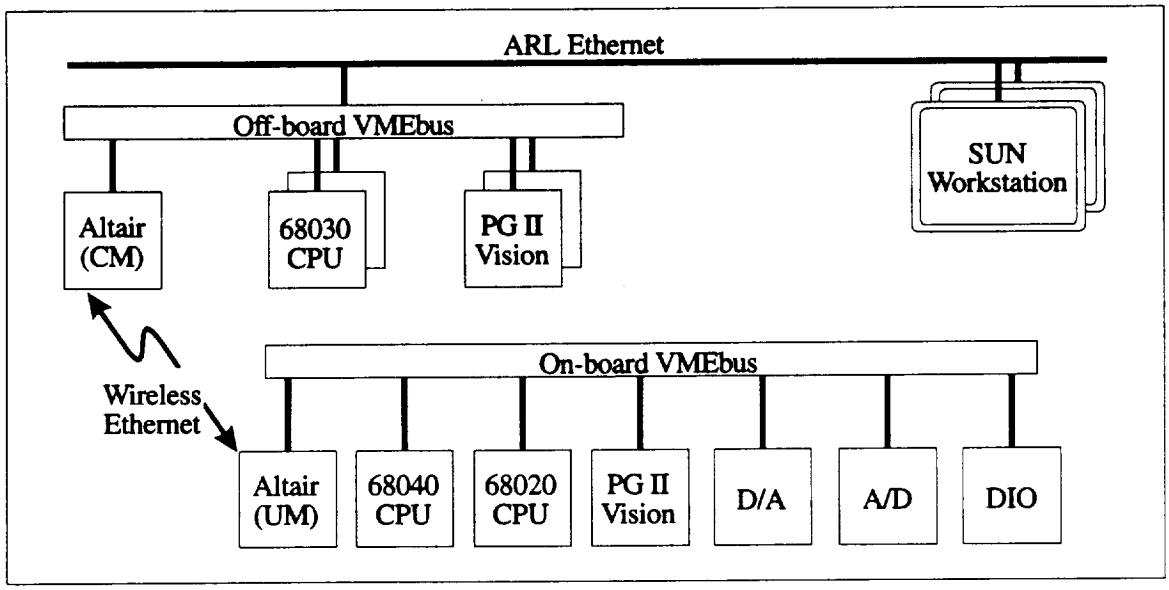


Figure 8.4: **Computer Subsystem**

Each of the vision-processing computers communicate with a Point Grabber II board.

The wireless Ethernet is the Motorola ALTAIR system. It consists of a Command Module and multiple User Modules. Each module appears as a "node" on the Ethernet, and communicate with each other at 19GHz microwave frequencies with an effective throughput of over 3Mbits/sec. Each module contains a six-sector antenna, and each continually monitors all signal paths to determine the best signal and to reject multipath signals, resulting in a very robust Ethernet connection in the laboratory setting. The Command Module is directly connected to the laboratory network, and a User Module is placed on the space robot. Additional User Modules are placed on other space robot experiments in the same laboratory[10].

The Xycom XVME590 16 differential channel 12-bit Analog-to-Digital (A/D) and the Xycom XVME595 4 channel 12 bit Digital-to-Analog (D/A) boards translate sensor input and actuator output to and from digital format used by the computer. A Xycom XVME290 digital input/output board furnishes a user-settable clock that is used for the sample clock. It also operates the on-off thrusters, the grippers, and the sensor multiplexers so the computer may monitor the battery voltages, gas tank pressure, and regulated pressure.

A SUN workstation completes hardware requirements for the space robot experiment. The workstation is the software development center for the space robot. All the run-time code is written and compiled on the workstation before sending it to the real-time computers. The workstation also runs a graphical user interface for interacting with the space robot. The workstation additionally is used for data collection and analyses. Because the space robot is hooked up to the Ethernet, any workstation—or multiple workstations—may be used for these purpose.

8.3 Software Architecture

The software architecture is developed to enable the user to interact with the space robot at a high task level, such as directing the robot to chase down and capture a free-flying object with a click of a button. The software is divided into three major areas; the graphical user interface, the strategic controller, and the dynamic controller. The graphical user interface runs on the workstations and communicates with the real-time computer systems to acquire position and orientation information for display and to send commands to the space robot. The strategic controller and the dynamic controller both run on the main real-time computer on the space robot. The strategic controller takes user inputs and schedules the necessary controller mode changes to carry out the task; the mode changes may occur at set time intervals or upon satisfying some conditions based on sensor inputs. For example, when the end effectors have been tracking the payload object grip ports to within an error tolerance, the grippers are engaged, and the control switches from end-point control to object control. The dynamic controller includes all the inverse-dynamics, adaptive update, Jacobian, and trajectory calculations. Both the strategic controller and the dynamic controller are implemented in the Controlshell real-time software environment developed by Stanley Schneider of ARL[11] [28, 34, 43].

8.3.1 Graphical User Interface

The graphical user interface (GUI) is the user's link to the space robot. It displays the global position of the robot and other objects of interest, and it allows the user to issue simple commands such as "move," "capture," and "release." This furnishes the desired high-level interface and disencumbers the operator from the burden of performing continuous hand-in-glove-type teleoperations.

[11] Extensions to Controlshell have been implemented by Marc Ullman and the author.

The graphical user interface communicates directly with the global-vision-system computers to acquire the global positions of the robot and the payload objects for display. Since the user is not in the high-speed feedback loop, the position display need not occur at high speed; high-performance control is immune to communications delays[12] and graphical display delays.

Figure 8.5 shows a portion of the graphical user interface. The space robot is indicated by the circle with the cross; the arrow on the cross shows the robot's heading. The payload object is similarly

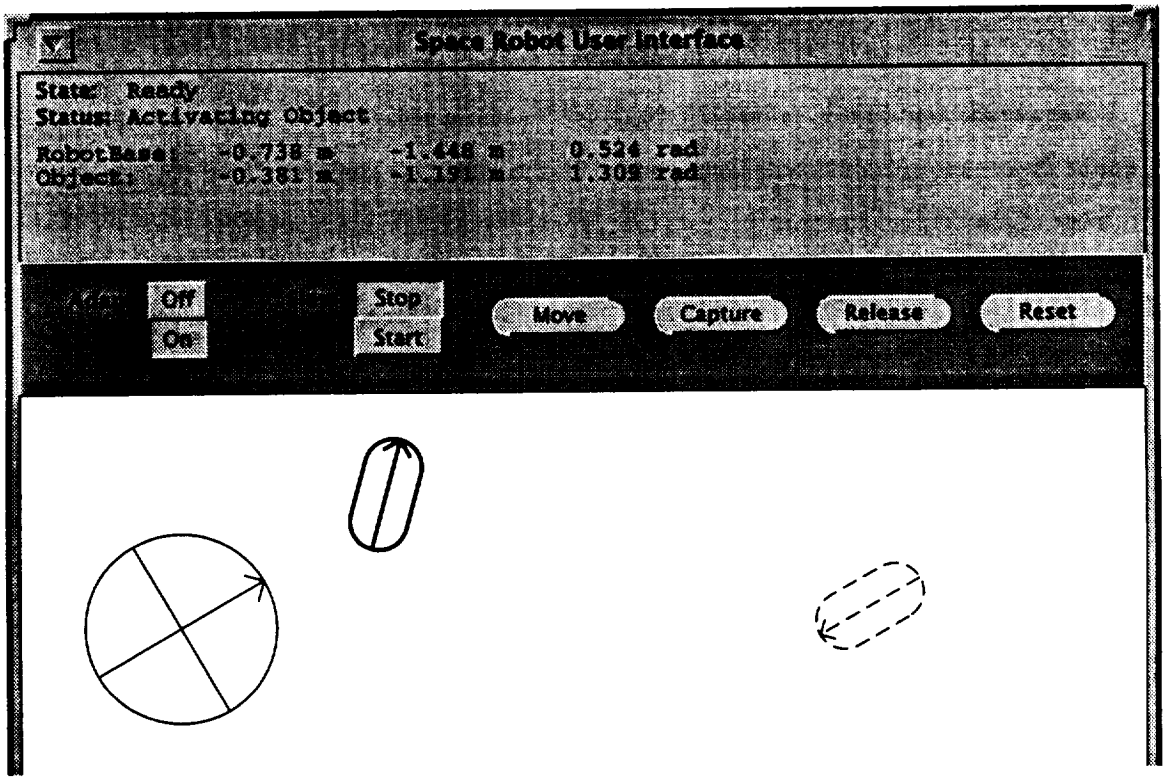


Figure 8.5: **Graphical User Interface**

The space robot is represented by the circle with the cross-hairs. The payload object is represented by the oval with cross-hairs that has the solid outline. The ghost image of the object, represented by the dashed outline, shows the desired payload position. The command buttons for "move," "capture," "release," and "reset" appear in the top portion of the interface. Adaptation may also be enabled through a command button. The manipulators are not shown, because their configurations are not essential for the user.

displayed, but with an oval pattern. The solid outlines indicate the actual positions of the objects, while a dotted outline—known as the "ghost image"—represents the desired position and orientation of the

[12]Communications delay is insignificant in the laboratory, but is an issue with real robots in space.

object. The human operator moves this ghost image employing a computer mouse. When the ghost image has been placed at the desired location and orientation, the operator clicks on the "move" button to initiate the action. If the object being moved is the space robot, the control system will move the robot base to the desired location using thrusters. If the object to be moved is the payload, the robot will first capture the payload, then move the payload to the desired location. The operator may also issue separate commands to "capture" and "move" the payload. Clicking on "release" directs the space robot to release a payload, and clicking on "reset" resets the robot to a known state. Additionally, the user may enable or disable adaptation through the graphical user interface.

Because the user directly commands the position and orientation of the robot or payload, he need not be concerned with the manipulator configurations; *task*-space control and the strategic controller manage all the detailed arm manipulation maneuvers. Accordingly, the manipulators are not displayed in the user interface.

8.3.2 Strategic Controller

The strategic controller is the heart of the controls system and is implemented using state-table programming [32, 43]. A "state" represents the current robot activity, and is typically one step in the execution of a requested task. State transitions are triggered by external stimuli; each state determines which stimuli to monitor. When a stimulus occurs, the strategic controller executes the associate transition routine, and the result from the transition routine determines the next state.

A stimulus may be an *edge-triggered* or a *persistent* event. Examples of *edge-triggered* stimuli are the expiration of a timer and the completion of a trajectory. These stimuli are active only for an instant. If they happen when they are not being monitored by the current state, they are lost. *Persistent* stimuli typically indicate status, such as the found/lost status of an object or the up/down status of the grippers. By being persistent, a state need not be monitoring the stimulus when it changes; it also enables a state to combine stimuli to form a new one, for example "GrippersDown" is formed by "RightGripperDown AND LeftGripperDown." If "RightGripperDown" and "LeftGripperDown" were both *edge-triggered*, such a combination would not be possible without forming intermediate states.

State-table programming can be illustrated graphically by a state-transition graph, such as the example in Figure 8.6. This example is a simplified state-transition graph for the capture of a payload. The robot starts in the "Ready" state. When it receives the "Capture" command from the user interface,

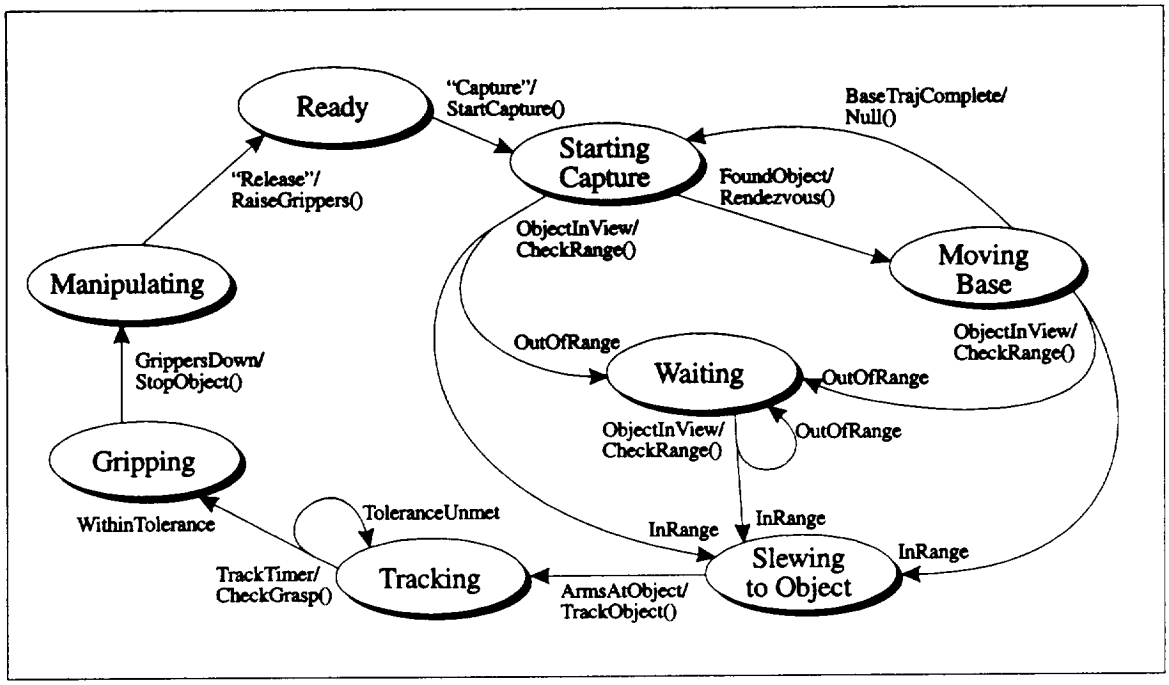


Figure 8.6: **Sample State Transition Graph**

This state transition graph example illustrates the basic components of the strategic controller in performing a payload capture.

it executes the "StartCapture()" transition routine, and ends up in the "Starting Capture" state. If the object is not within the view of the local camera, but it has been found by the global positioning system, the robot plots a trajectory for the robot base to *rendezvous* with the object and enters the "Moving Base" state. If when the base trajectory has been completed but the object has not come into view yet, the robot re-enters the "Starting Capture" state to plot a new base trajectory. When the object comes into view of the local camera in either the "Starting Capture" or "Moving Base" state, the robot runs the "CheckRange()" routine to determine if the object is within the reach of the arms; if out of range, the robot enters the "Waiting" state; otherwise, it proceeds to the "Slewing to Object" state. While in "Waiting", the robot continually checks for the object to come into reach before transitioning to "Slewing to Object."

"Slewing to Object" places the manipulators under endpoint control to command the arms to follow a trajectory to intercept the grip ports on the payload. When both grippers reach the grip ports, "ArmsAtObject" triggers the transition to "Tracking", where the grippers must track the payload

grip ports for a set period of time to ensure that the grip tolerance is met. If at the end of the time period—indicated by the "TrackTimer" stimulus—the tolerance is not met, the manipulators must continue to track the grip port; otherwise, the robot lowers the grippers. When both grippers are down, the robot employs object control[13] to execute the "StopObject()" transition routine and enters the "Manipulating" state. If the robot receives the "Release" command while "Manipulating", it raises the grippers and returns to the "Ready" state to await further commands.

Error handling is easily implemented with a state-table programming. In any state before the payload has been grasped, an "ObjectLost" stimulus from the vision system can bring the robot into an "Error" state[14], and eventually back to the "Ready" state. If, while manipulating the object, the "GripToleranceExceeded" stimulus occurs—indicating that the robot may have lost grip on the object—the robot may release the object and attempt to reacquire it.

Similar state transition graphs have been developed and implemented[15] for processing other user commands to move the robot base, move the payload, and enable adaptive control.

8.3.3 Controlshell

Controlshell is a software framework developed by Stan Schneider at ARL that aids the development of real-time control systems. Controlshell enables the user to develop a multiple-processor hierarchical control system in a modular fashion. It directly supports state-table programming utilized by the strategic controller. At the lower level, Controlshell incorporates user-defined components for performing any number of control-system tasks ranging from acquiring sensor values to computing the *task*-space control and adaptive update laws. Controlshell also allows definitions of complete configurations of components—configurations that comprise all the components that pertain to a control mode. These configurations may be swapped dynamically to effect control-mode changes under the direction of the strategic controller or the user. Please see [34, 43, 28] for more details on Controlshell.

The *task*-space adaptive controller is implemented at this software level. Different Jacobian components are defined to correspond with the different *task*-space control objectives representing different

[13]Object control, endpoint control, and joint control are all implemented with the *task*-space adaptive controller.

[14]When the object is being grasped, the robot may infer the object location from its joint sensors and knowledge of the grip port locations on the payload, so the vision system losing the object should not cause the robot to abort manipulation. The grip is also monitored to determine if the robot *is* grasping the payload.

[15]The state-transition tables were originally developed by Marc Ullman of ARL [43]. The author made modifications and added capabilities to handle adaptive control.

control modes. Appropriate torque-mapping components are also defined. There is, however, only one component for the inverse-dynamics and one for the adaptive update calculations.

As a side note, the real-time computers run the VxWorks real-time operating system from Wind River Systems. In addition to providing facilities for real-time activities, the operating system supports networked file systems, allowing each computer to download its run-time code directly from the laboratory file systems via the high-speed Ethernet connections. Controlshell runs on top of VxWorks.

8.4 Payload Subsystems

Figure 8.7 depicts the two free-flying payload objects used to investigate the performance of the adaptive controller. They possess inertia properties that differ by an order of magnitude. These differences are

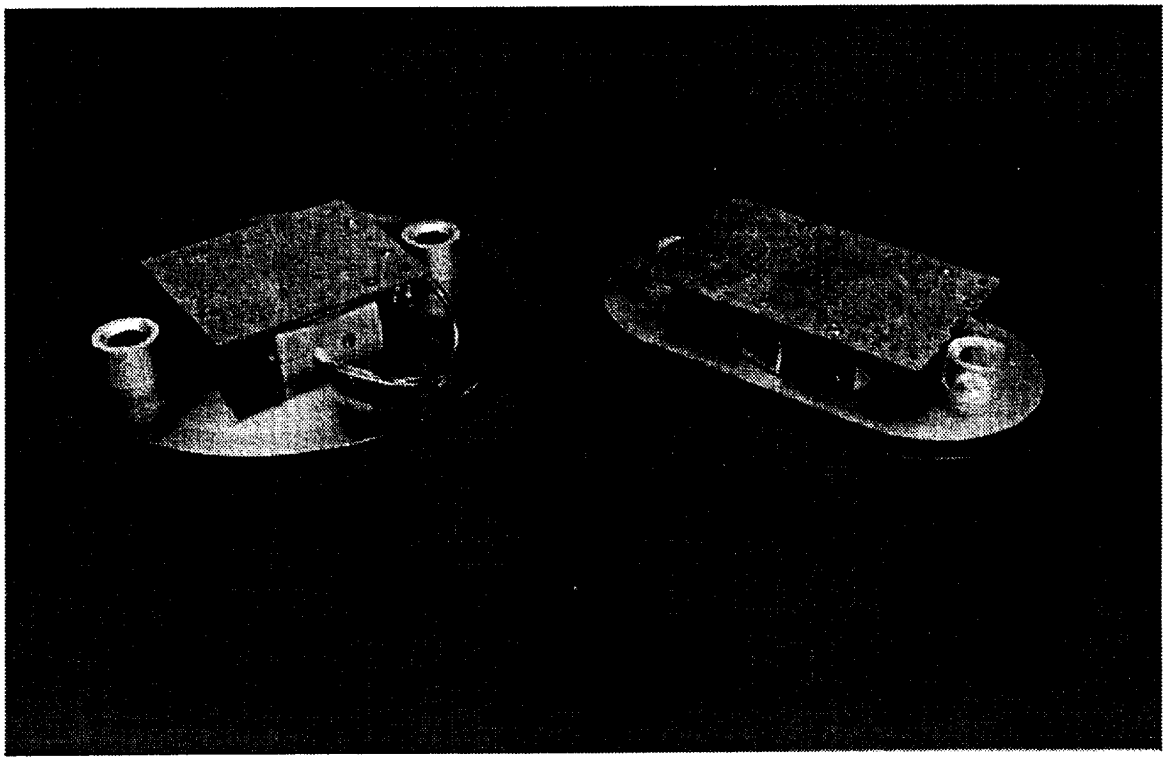


Figure 8.7: **Free-Flying Payload Objects**

The free-flying payload objects have mass properties that differ by an order of magnitude, but they appear identical to the vision system and the space robot, because the LED patterns and grip port locations are identical.

enough to demonstrate significant deterioration in controller performance were adaptation not used.

Both payload objects float employing battery-operated aquarium pumps. Each top plate includes three infra-red LED's for identification and tracking by the vision systems. The patterns are identical for both objects, such that the vision system and space robot cannot know which object the robot is to capture and manipulate. The grip ports on both payloads also are located identically. The smaller object is made of honey-combed aluminum, and its mass is 1.0kg and its moment of inertia is 0.007kg-m^2. The larger payload is made of solid stainless steel and masses 8.9kg; its moment of inertia is .1kg-m^2.

Chapter 9

Implementation and Experimental Results

The new *task*-space adaptive controller was implemented on the Multi-Manipulator Free-Flying Space Robot. This controller, coupled with the strategic controller described in Chapter 8, allows the space robot to execute complicated control sequences—including the chase, capture, and placement of free-flying objects—with adaptation available throughout all control-modes. This chapter documents experimental results verifying the performance of the adaptive control for two of control modes, or *tasks*: object control and endpoint control.

Payload adaptation offers the most dramatic results, since changing payloads causes step changes in the system parameters; and the adaptive controller must respond quickly to these step changes to maintain good performance. Experimental results of adaptation to two payloads with an order-of-magnitude difference in inertial parameters is presented. They show the poor performance of a nonadaptive controller when the controller is given the wrong set of payload inertial parameters. They also show the improvement when adaptation is activated. The results suggest that adaptive control can deliver performance equal to or even better than a nonadaptive controller using *nominal* payload inertial parameters.

This chapter also shows the capability of the *task*-space adaptive controller to adapt to *robot* parameters. The inertial parameters of the right manipulator are set to zero, and endpoint control is enabled. The results indicate that the adaptive controller can also adapt to large changes in the robot parameters and thereby greatly reduce the endpoint trajectory errors.

9.1 System Model

Figure 9.1 shows the schematic used to model the Multi-Manipulator Free-Flying Space Robot. The

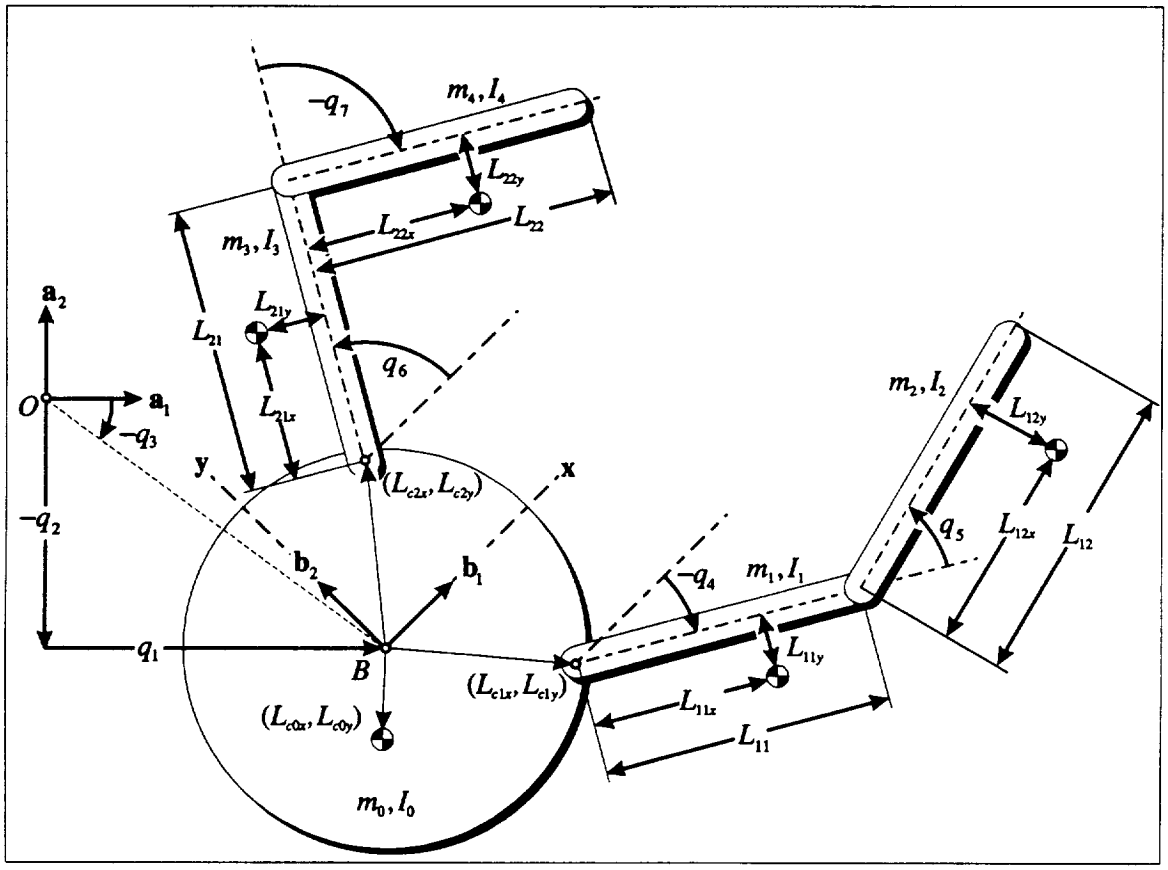


Figure 9.1: **Multi-Manipulator Free-Flying Space Robot Schematic**

The space robot consists of five rigid bodies. Each body i has mass m_i and moment of inertia I_i. The center of mass of the base is offset from the geometric center of the base by (L_{c0x}, L_{c0y}). The shoulder of each manipulator j is located at (L_{cjx}, L_{cjy}), measured from the base center. The upper link of each manipulator j is L_{j1} long, and its center of mass is located at (L_{j1x}, L_{j1y}), measured from the shoulder. The lower link of each manipulator j is L_{j2} long, and its center of mass is located at (L_{j2x}, L_{j2y}), measured from the elbow. The base-relative coordinate system is fixed in the base frame and is centered in the base; the x*-axis is aligned with the unit-vector* $\mathbf{b}_1$*, and the* y*-axis is aligned with the unit-vector* $\mathbf{b}_2$.

space robot is a seven-degree-of-freedom system, consisting of five rigid bodies. The free-flying base has three degrees of freedom—two in translation and one in rotation, and each manipulator link has one rotational degree of freedom. The mutually perpendicular unit-vectors, $\mathbf{b}_1$ and $\mathbf{b}_2$, are fixed in the

robot-base frame. The base-relative coordinate system is fixed in the base, and its center coincides with the center of the base; the body-fixed x-axis is aligned with $\mathbf{b}_1$, and the y-axis is aligned with $\mathbf{b}_2$.

The payload object is a single rigid body, possessing three degrees of freedom. Figure 9.2 shows the schematic used to model the payload. The space robot and the payload are combined to form a complete system model used by the *task*-space adaptive controller.

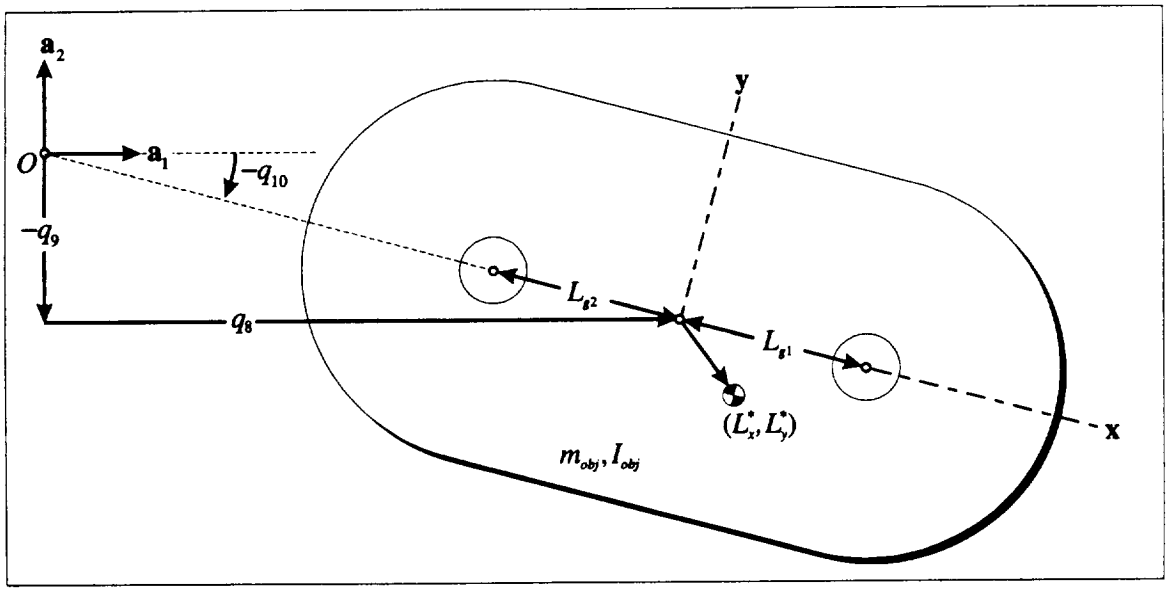


Figure 9.2: **Payload Object Schematic**

The payload is a single rigid body with three degrees of freedom. It has mass m_{obj} and moment of inertia I_{obj}. The center of mass is located at (L_x^*, L_y^*), measured from the geometric center of the object. The grip ports are located at L_{g1} and L_{g2} along the center line. The body-relative coordinate system is fixed in the payload, and the x- and y-axes are as indicated.

The robot may be modelled with sixteen (16) parameters, and the payload modelled with another

four, giving the twenty-term system parameter vector[1]:

$$\theta = \begin{bmatrix} m_0 + m_1 + m_2 + m_3 + m_4 \\ m_0 L_{c0x} + \left(m_1 + m_2\right) L_{c1x} + \left(m_3 + m_4\right) L_{c2x} \\ m_0 L_{c0y} + \left(m_1 + m_2\right) L_{c1y} + \left(m_3 + m_4\right) L_{c2y} \\ m_1 L_{11x} + m_2 L_{11} \\ m_1 L_{11y} \\ m_2 L_{12x} \\ m_2 L_{12y} \\ m_3 L_{21x} + m_2 L_{21} \\ m_3 L_{21y} \\ m_4 L_{22x} \\ m_4 L_{22y} \\ m_0 L_{c0}^2 + \left(m_1 + m_2\right) L_{c1}^2 + \left(m_3 + m_4\right) L_{c2}^2 + I_0 \\ m_1 \left(L_{11x}^2 + L_{11y}^2\right) + m_2 L_{11}^2 + I_1 \\ m_2 \left(L_{12x}^2 + L_{12y}^2\right) + I_2 \\ m_3 \left(L_{21x}^2 + L_{21y}^2\right) + m_4 L_{21}^2 + I_3 \\ m_4 \left(L_{22x}^2 + L_{22y}^2\right) + I_4 \\ m_{obj} \\ m_{obj} L_x^* \\ m_{obj} L_y^* \\ m_{obj} \left(L_x^{*2} + L_x^{*2}\right) + I_{obj} \end{bmatrix} \tag{9.1}$$

Payload adaptation is demonstrated utilizing base-relative control. This is an important capability for a free-flying robot, because local sensors can provide high-bandwidth, high-resolution sensing that may not be available from global sensing systems. A typical manipulation task is to extract or insert the payload from or into a mating part. It is important for the local sensing system to sense both the payload and the mating "port" during these maneuvers.

To model this situation, a separate "port" object is placed in the field of view of the local vision system, and the robot is directed to perform slews to, and regulate at positions fixed in the reference frame of the "port", as illustrated in Figure 9.3. Because the space robot base may move during the slews, the "port" is *not* fixed in the robot's reference frame. With relatively low feedback on the space robot base position and orientation, the base is free to move within a bounded area before the on-off thrusters fire. This can save precious fuel when performing local manipulation.

Experimental results are shown for handling of the small and large payloads with and without adaptation. Note that only the last four parameters need to be updated for payload adaptation.

Adaptation to robot parameters is demonstrated with endpoint control: the right manipulator endpoint is commanded to follow trajectories in the robot-base reference frame. The inertial parameters of the manipulator are set to zero to compare the performance with and without adaptation. All

[1]The derivation is left to the reader.

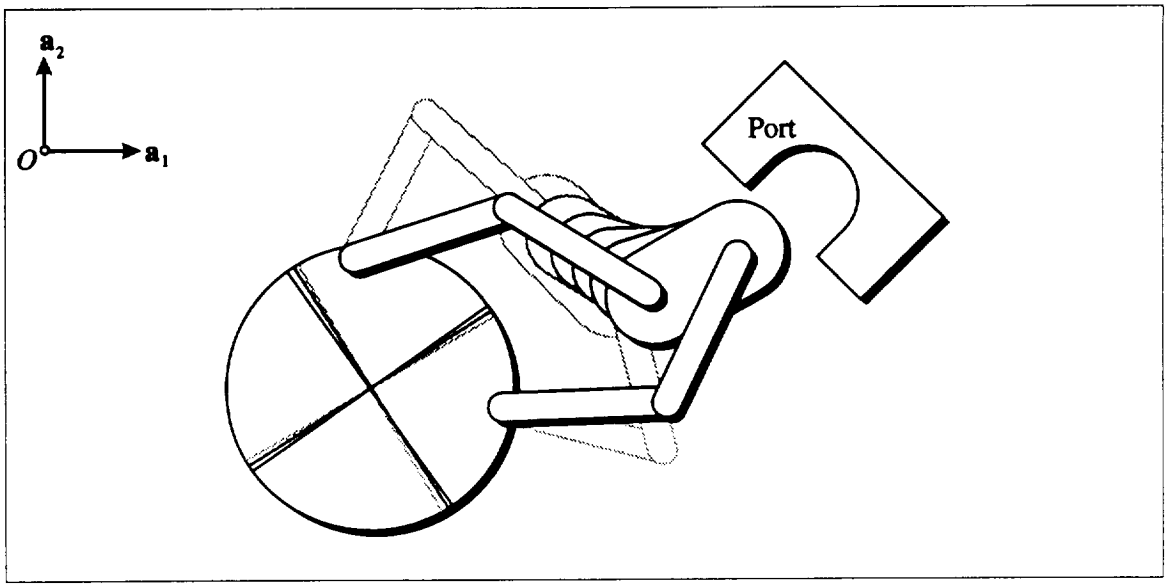


Figure 9.3: **Payload Slews Relative to Mating "Port"**

This schematic shows a typical slew that the space robot performs. The payload is directed to follow a trajectory in the "port" reference frame, in the presence of robot base motions. The cross-hairs in the robot base indicate that the robot base may move during the slew.

the parameters containing the manipulator inertial parameters—m_1, m_2, I_1, and I_2—are adaptively updated.

9.2 Adaptation to Small Payload

The experimental results for the control of the small payload is separated in four sections: nonadaptive control using nominal payload parameters, adaptive control starting with nominal parameters, nonadaptive control starting with incorrect parameters, and adaptive control with incorrect parameters.

9.2.1 Nonadaptive Control with Nominal Payload Parameters

The results for the nonadaptive control using nominal payload parameters serve multiple purposes; it represents a baseline controller to which the adaptive controller is compared, and it serves to introduce the format of the data plots. The payload is commanded to follow back-and-forth slews, illustrated by Figure 9.4. This time-lapsed "overhead view" of the system is plotted from actual experimental data.

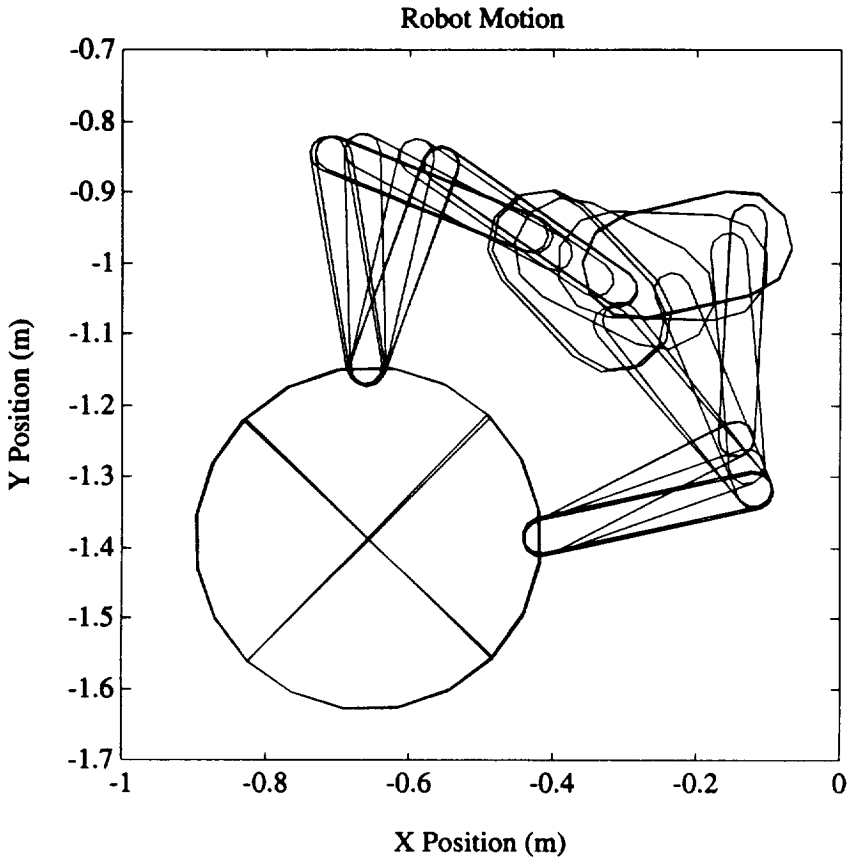


Figure 9.4: **Robot Slew from Actual Data**

This plot depicts a typical slew that the robot is commanded to follow. The "view" is taken from inertial space. The robot base is under control, but the control gains are adjusted to allow small base movements before the on-off thrusters fire; close inspection of the plot shows that the base did rotate several degrees.

Figure 9.5 displays the plots for the actual and desired position and orientation of the payload in the "port" reference frame. The lower right plot shows the Cartesian path that the payload followed during the slews. These time-histories show that the controller performed reasonably well in trajectory-tracking. The steady-state offsets are caused by spring forces from tubing inside the manipulators. Integral control reduces these errors, but it has been disabled to compare the performance of the basic *task*-space controller with and without adaptation.

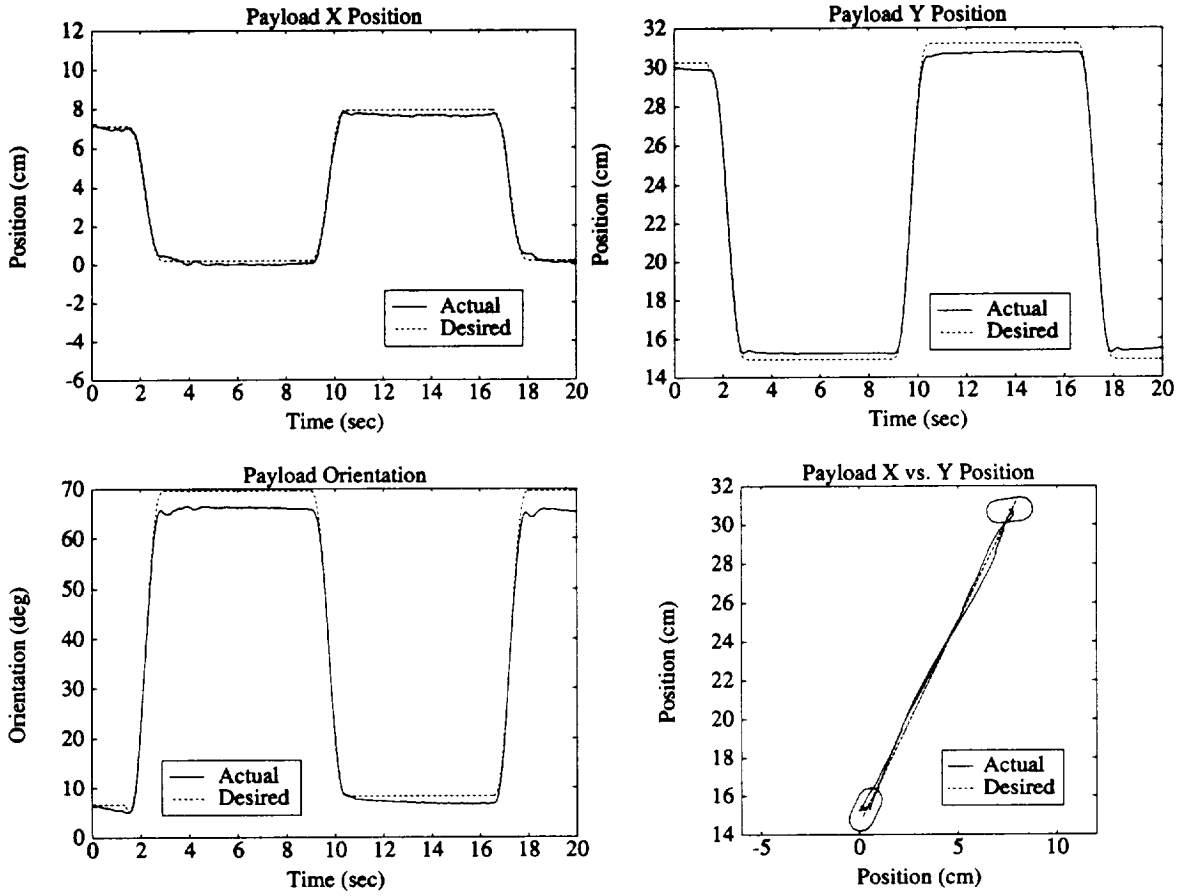


Figure 9.5: **Small-Payload Trajectories—Baseline Nonadaptive Control**

The actual and desired payload trajectories measured in the "port"-fixed reference frame, which is fixed in inertial space for the series of experiments presented here. The solid lines represent the actual measured payload trajectories of the payload's geometric center and the dashed lines represent the desired trajectories. The Cartesian trajectories are in the top two plots, and its accompanying orientation is in the lower left plot. The lower right plot shows the "X vs. Y" trajectory, representing an "overhead" view of the path traced out by the center of the payload, as seen from a reference frame fixed to the "port". The oval icons at the ends of the slews show the payload orientations at those locations, but the icons do not *depict the actual payload—the icons are* much *smaller than the actual payload.*

9.2.2 Adaptive Control with Nominal Payload Parameters

This section presents results for enabling adaptive control when the payload parameters are at their nominal values. Doing so allows comparisons of the adaptive controller performance and the baseline nonadaptive controller under near ideal conditions. It also can show how the parameters will eventually

converge.

Figure 9.6 demonstrates that adaptation does not deteriorate the trajectory-tracking performance (compare with Figure 9.5). The time-histories of the payload-parameter estimates are shown in

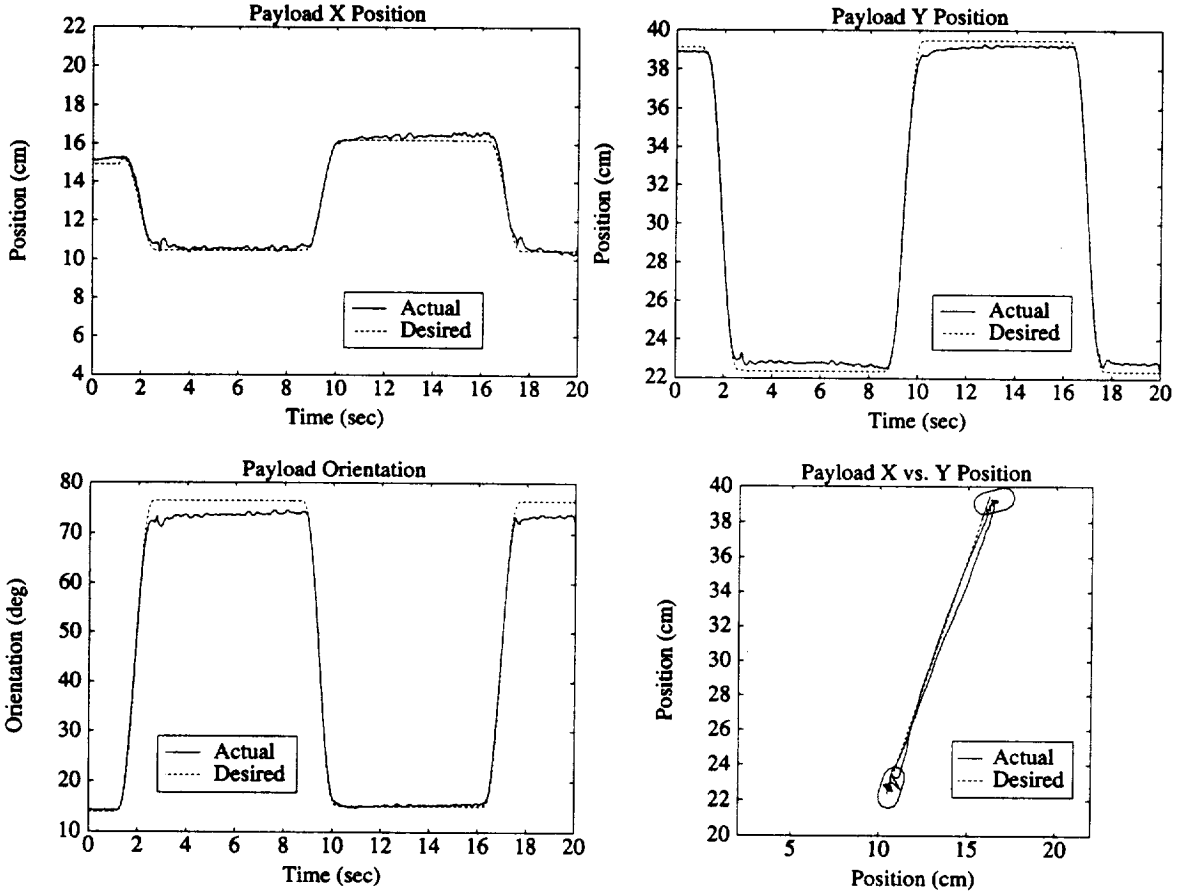


Figure 9.6: **Small-Payload Trajectories—Adaptive Control Starting with Nominal Values for Parameter Estimates**

These actual and desired payload trajectories are measured in the "port"-fixed reference frame. They show the results for the adaptive controller starting with the nominal payload parameter estimates. The time-histories show no significant difference in performance when compared to the baseline nonadaptive controller. The oval icons in the lower-right plot show the payload orientation at those locations.

Figure 9.7. The payload mass directly corresponds to the parameter θ_{17} (see Equation (9.1)); the center-of-mass locations and the moment of inertia may be solved using the last four parameters of $\boldsymbol{\theta}$. The results show that the mass and center-of-mass estimates do hover around their nominal values; but the moment of inertia has a larger relative variance around its nominal value. A possible explanation is

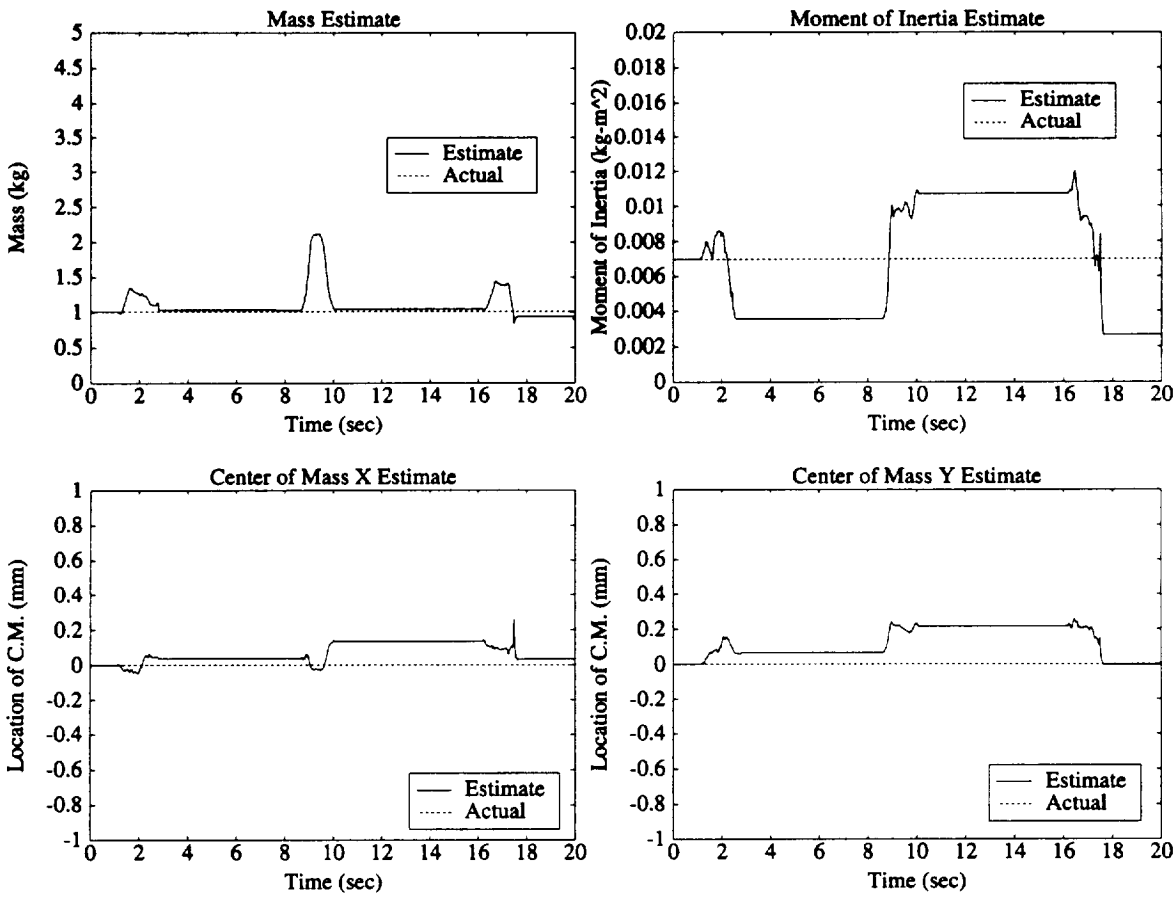


Figure 9.7: **Small-Payload Parameter Estimates—Adaptive Control Starting with Nominal Values for Parameter Estimates**

These plots show the time-histories of the parameter estimates when they are initially given the nominal values, and then the payload is moved as in Figure 9.5 and Figure 9.6.

that the spring forces in the arms are masking the inertial effects of the payload, since the true moment of inertia of the payload is small (.007kg-m^2). Nevertheless, the sharp changes in moment-of-inertia estimate do not affect the trajectory tracking performance.

9.2.3 Nonadaptive Control with Incorrect Payload Parameters

The real advantages of adaptive control are evinced when the controller is supplied incorrect parameter estimates. This section presents results for the nonadaptive controller controlling the small payload when given the payload parameters of the larger payload. Figure 9.8 shows the payload trajectories.

Because the controller believes the payload to be an order of magnitude more massive than it is, the

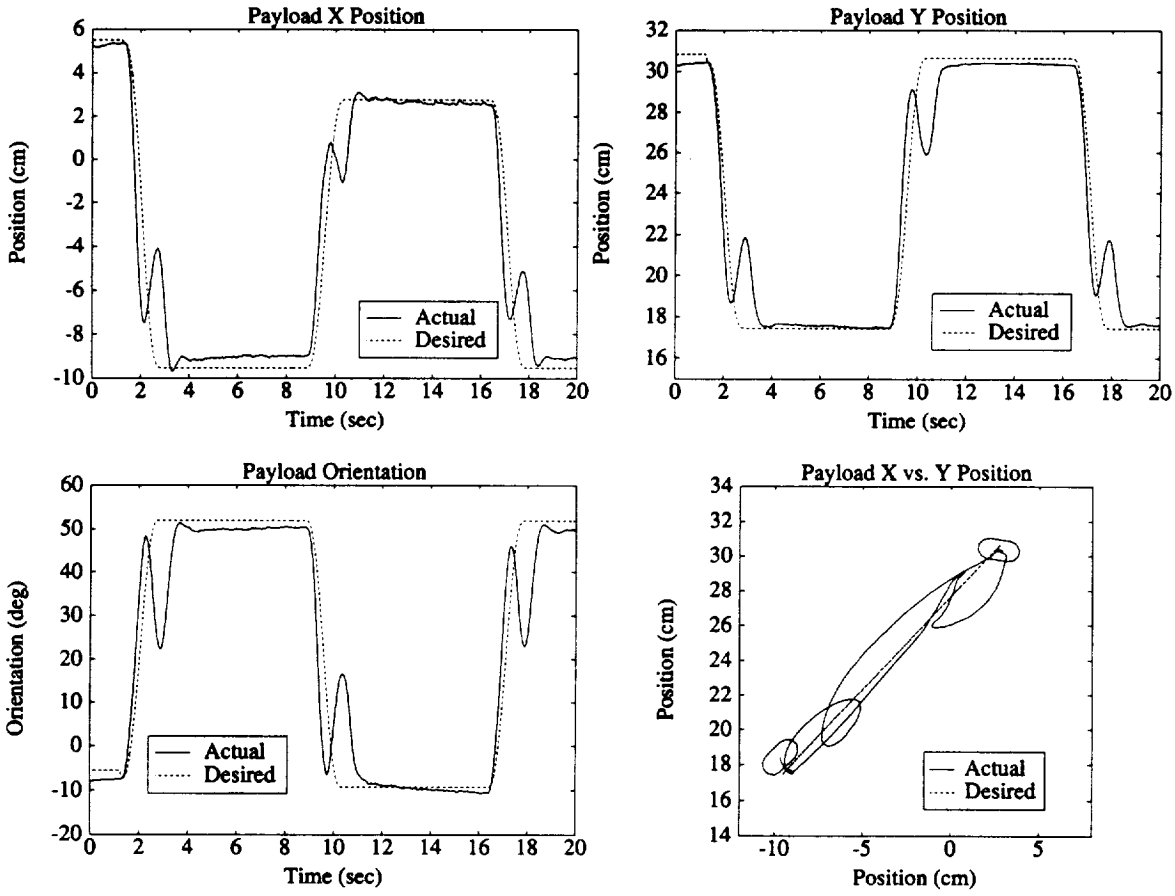


Figure 9.8: **Small-Payload Trajectories—Nonadaptive Control Starting with Incorrect Values for Parameter Estimates**

The actual and desired payload trajectories, measured in the "port"-fixed reference frame. They show the results for the nonadaptive *controller actually controlling the smaller payload, but starting with the larger-payload values for its parameters. The time history shows that this is an unacceptable controller (it skirts instability).*

controller requests too much actuator effort to initiate each slew. The plots demonstrate this by showing that actual trajectories lead the desired trajectories at the start of each slew. The feedback portion of the controller finally dominates toward the end of each slew to slow the payload, but causes large reversals in direction. This is not an acceptable controller. Many times this particular situation leads to violent instability.

9.2.4 Adaptive Control with Incorrect Payload Parameters

This time, adaptive control is enabled, with initial payload parameters set to that of the larger payload. Figure 9.9 shows a dramatic improvement in performance. The adaptive control updates the parameters

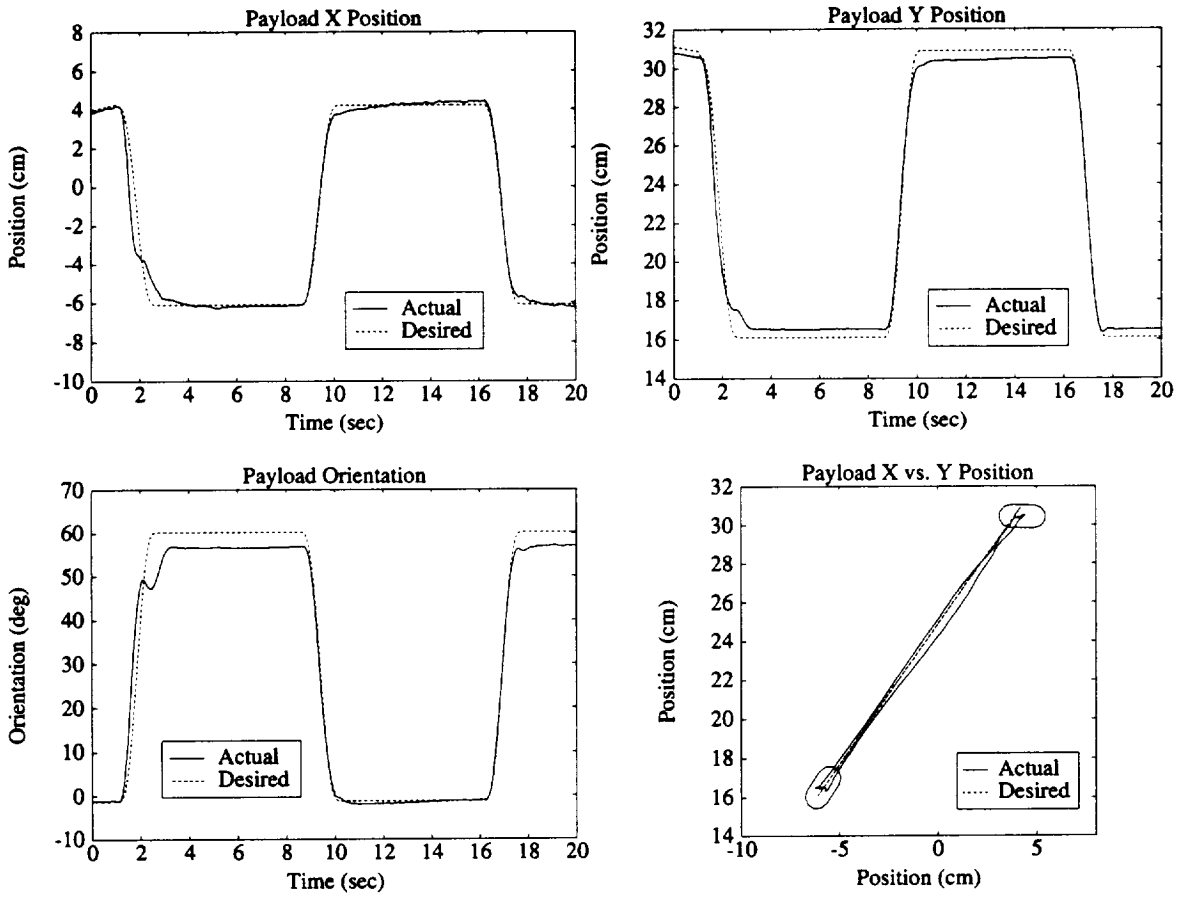


Figure 9.9: **Small-Payload Trajectories—Adaptive Control Starting with Incorrect Values for Parameter Estimates**

The actual and desired payload trajectories, measured in the "port"-fixed reference frame, show the results for the adaptive controller actually controlling the smaller payload, but starting with the larger-payload values for its parameters. The adaptive control adapts fast enough to prevent significant overshoot by the end of the first slew, and effectively recovers the baseline controller performance quality by the second and third slews (compare with Figure 9.8).

quickly enough to prevent significant overshoot by the end of the first slew. By the second and third slews, the controller performance rivals that of the baseline nonadaptive controller of Figure 9.5.

Figure 9.10 displays the time-histories of the parameter estimates. These plots show the rapid

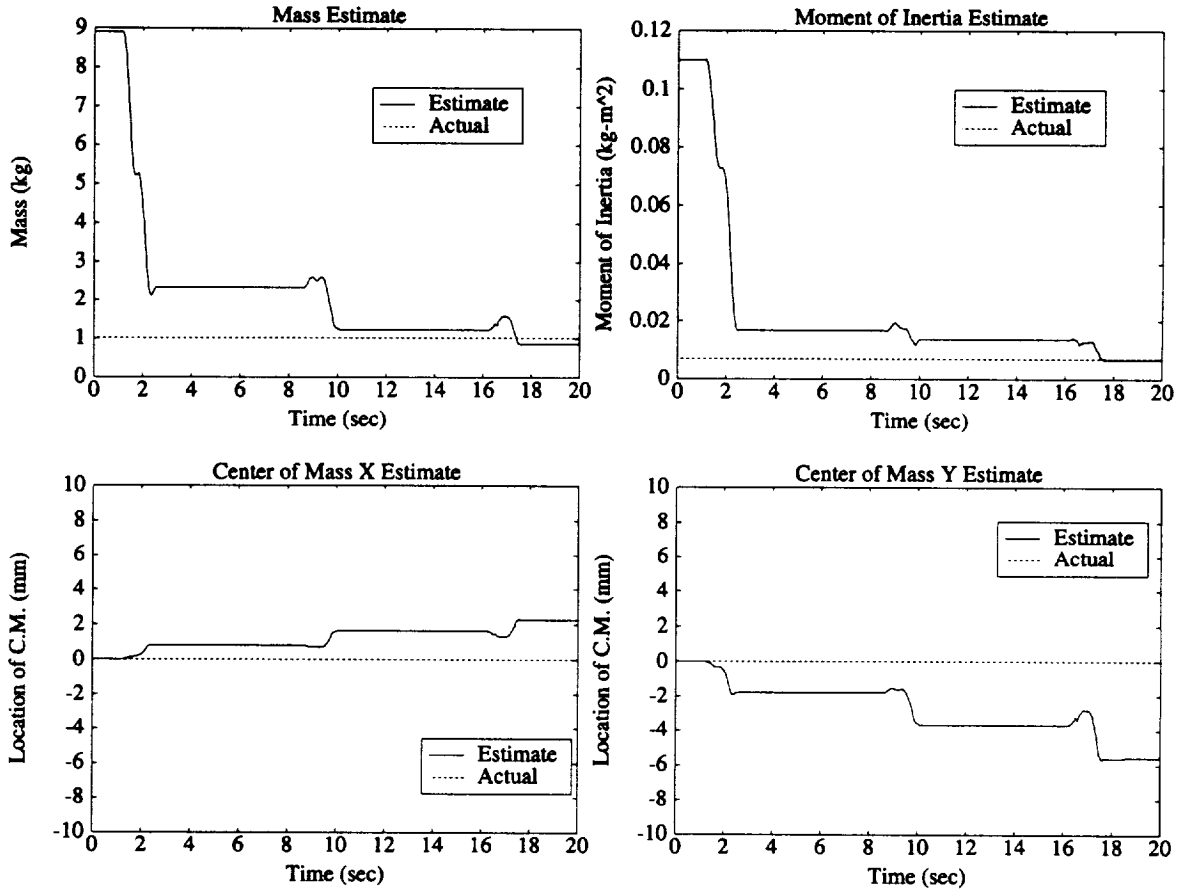


Figure 9.10: **Small-Payload Parameter Estimates—Adaptive Control Starting with Incorrect Values for Parameter Estimates**

These plots show the time histories of the parameter estimates when they are initially set to the parameters of the larger payload.

adaptation of the payload parameters. The mass and moment-of-inertia estimates converged rapidly toward their true values. The center-of-mass estimates, however, approached a location about 6mm from the nominal location. This demonstrates once again that, lacking a sufficiently exciting trajectory, trajectory-tracking convergence does not imply parameter convergence.

9.2.5 Small-Payload Control Summary

To summarize the results for control of the small payload, Figure 9.11 shows the trajectory-tracking errors of the baseline controller, the adaptive controller starting with the incorrect parameters, and

the nonadaptive controller utilizing the incorrect parameters. The plots show that the adaptive

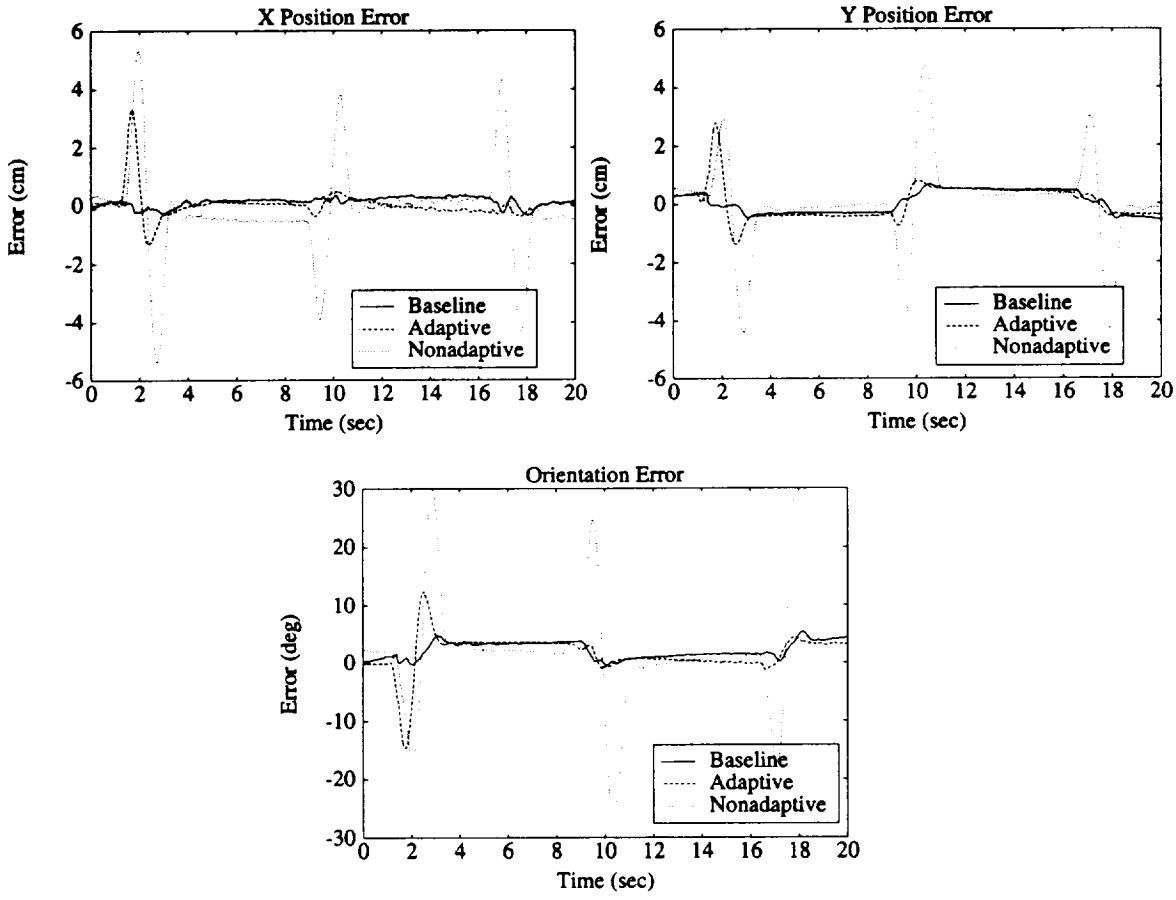


Figure 9.11: **Small-Payload Trajectory-Tracking Errors**

These plots show the time-histories of the trajectory-tracking errors of three controllers. The adaptive controller already performs much better than the nonadaptive controller on the first slew, and as well as the baseline controller—which is using the nominal parameters—after the first slew.

controller performed better than the nonadaptive controller, even though they started with the same set of incorrect payload parameters. By the second and third slews, the adaptive controller performs as well as the baseline controller.

9.3 Adaptation to Large Payload

Adaptation to the large payload duplicates the experiment in Section 9.2 using the larger payload. The first two show the results of the nonadaptive and adaptive controller starting with the nominal payload parameters (Figures 9.12 and 9.15). The nonadaptive controller represents the baseline controller. The last two sections show the two controllers starting with the smaller payload parameters (Figures 9.17 and 9.18).

9.3.1 Nonadaptive Controller with Nominal Payload Parameters

This section presents the baseline nonadaptive controller utilizing the nominal set of payload parameters. The same back-and-forth slews are performed. Figure 9.12 shows the time-histories of the actual and desired trajectories. The plots indicate that even the baseline controller exhibits some overshoot characteristics. The adaptive control analysis in the next section shows that the spring forces[2] in the manipulators are not likely candidates for causing the overshoots. Actuator saturation, however, is a likely culprit, as Figures 9.13 and 9.14 show. The top right plot in Figure 9.13 shows that the right elbow torque barely reaches saturation during the second half of each slew. The base force[3] in the X direction also saturates during each slew. Actuator saturation prevents the robot from providing enough actuation to slow the payload sufficiently at the end of each slew and causes the trajectory overshoots.

[2]The spring effects are caused by electrical and pneumatic conduits inside the manipulators.

[3]The base forces and torques in the plots correspond to requested values from the controller. The actual forces and torques are supplied by the on-off thrusters, utilizing optimal bang-off-bang thruster mappings.

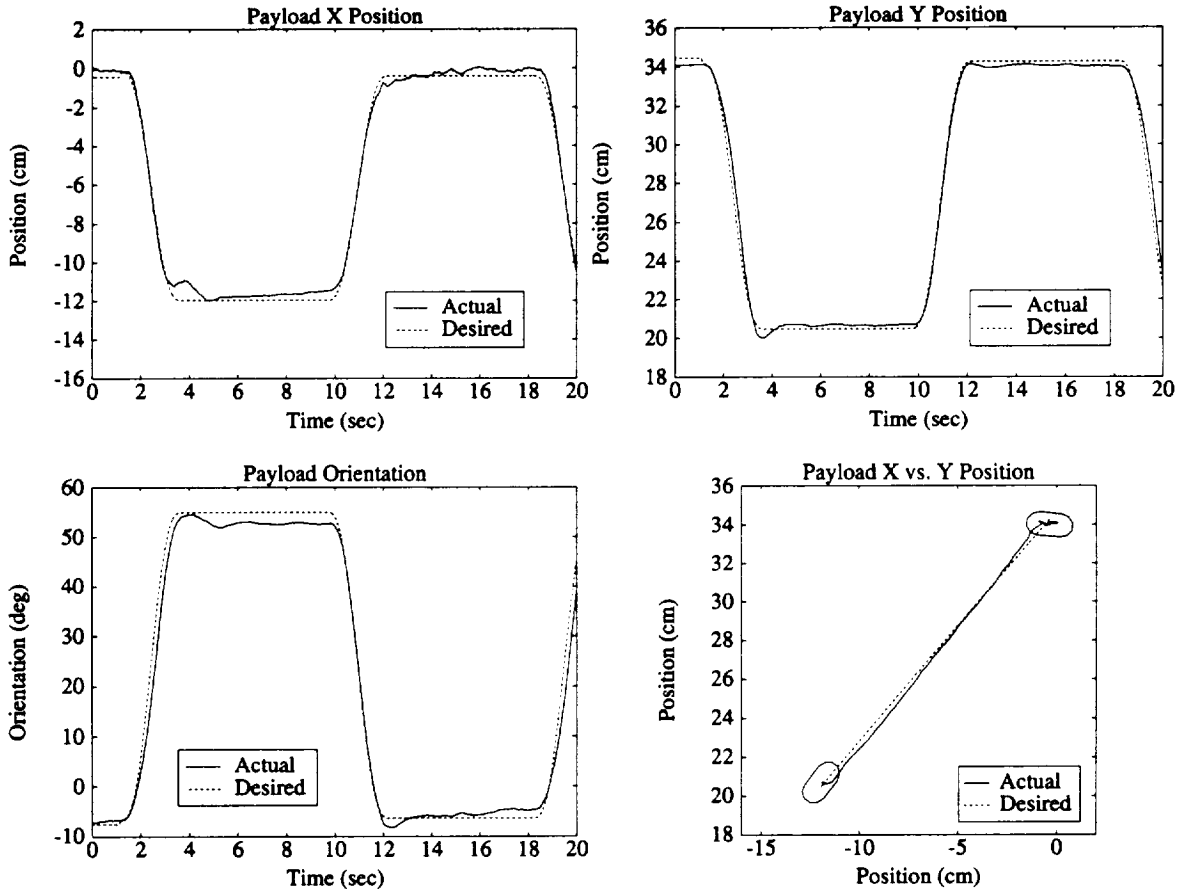


Figure 9.12: **Large-Payload Trajectories—Baseline Nonadaptive Control**

The actual and desired payload trajectories, measured in the "port"-fixed reference frame, show the results for the baseline nonadaptive *controller using the nominal values for payload parameters. The plots show that even the baseline controller has some overshoot.*

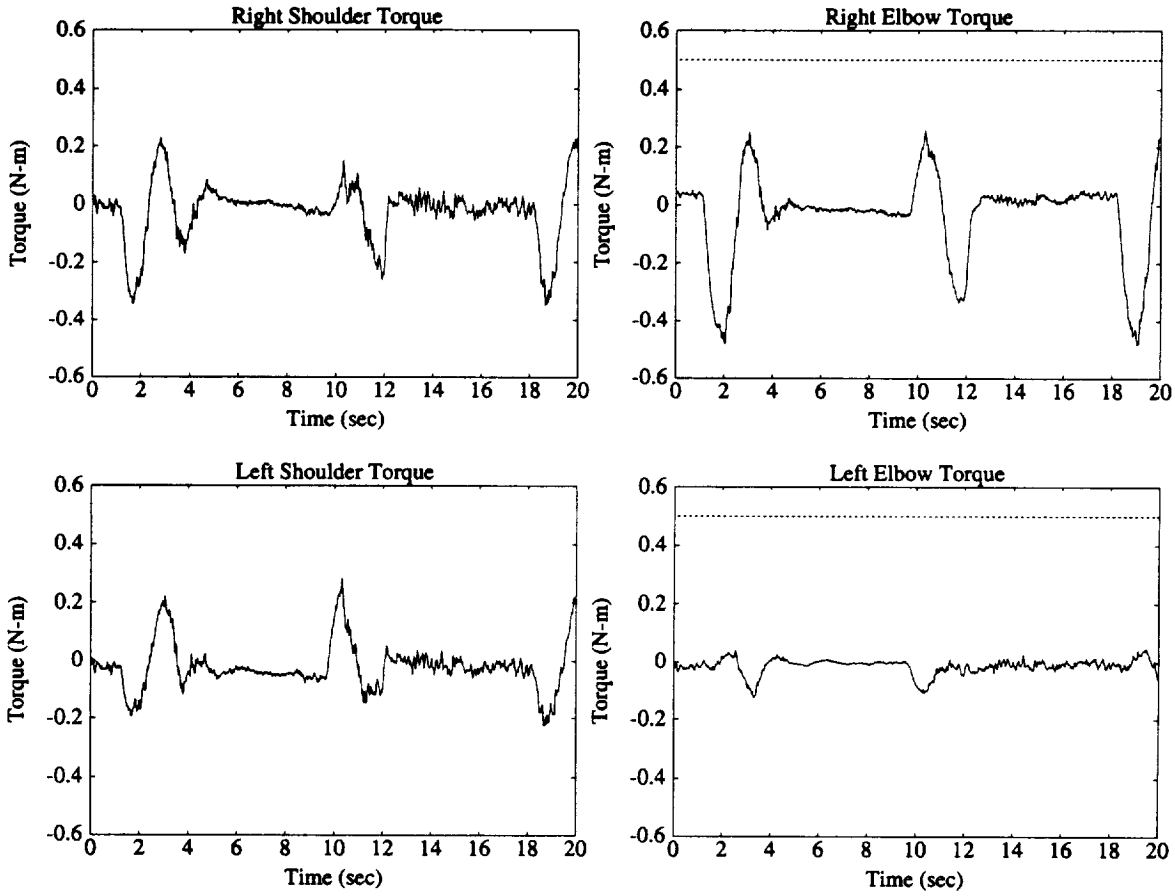


Figure 9.13: **Requested Motor Torques—Baseline Nonadaptive Control**

These are the motor torques requested by the controller. The elbow motors saturate at .5N-m, so the requested payload trajectory is right on the border of what the space robot can deliver. The top right plot shows that the right elbow motor almost reaches saturation during each slew.

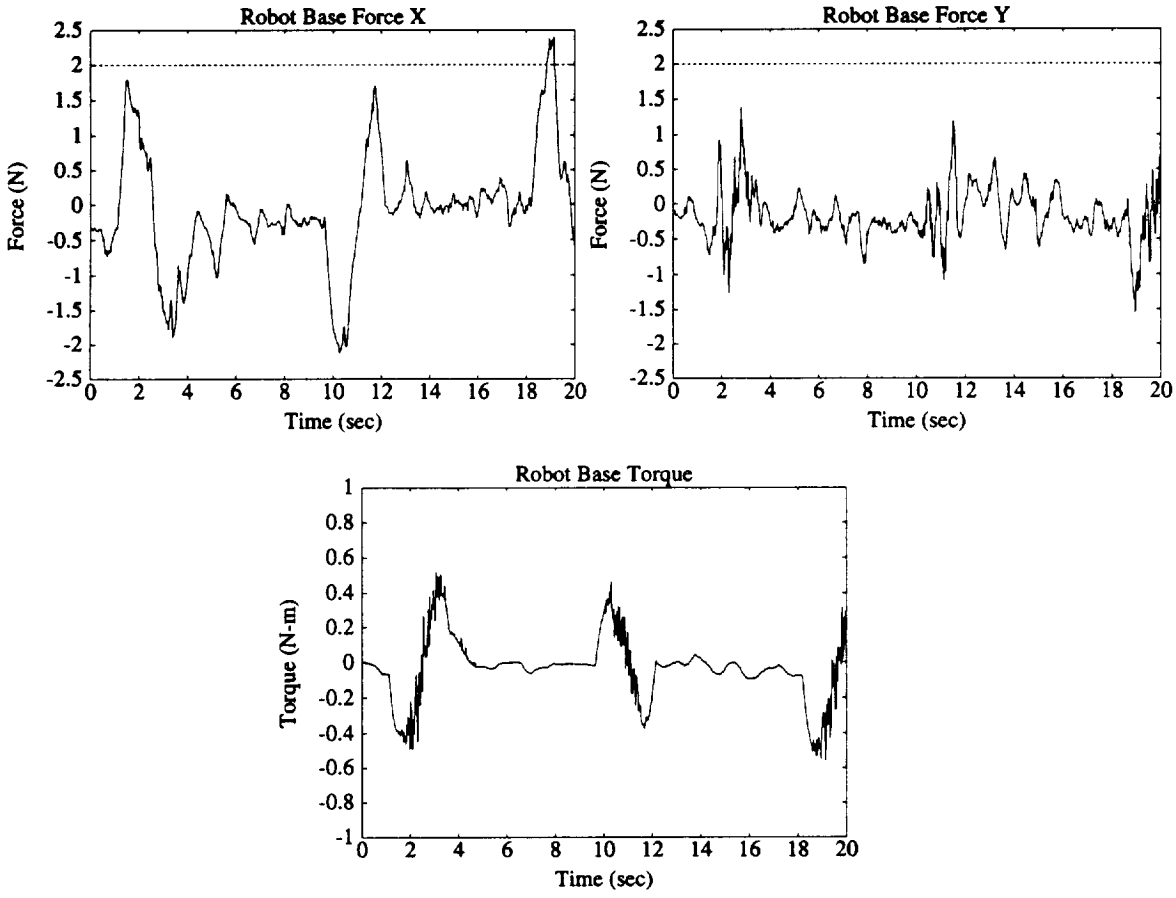


Figure 9.14: **Requested Base Forces and Torque—Baseline Nonadaptive Control**

These plots show the base forces and torques requested by the controller. The actual delivered forces and torques are derived from the optimal bang-off-bang thruster mappings. The base force in the X direction also saturates. The maximum torque capability is 1N-m, so the base torque, as shown in the bottom plot, is still far from saturating.

9.3.2 Adaptive Controller with Nominal Payload Parameters

This section shows once again that the adaptive controller, when starting with the nominal set of parameters, does not deteriorate the controller performance. In fact, the adaptive controller improves the orientation control, as Figure 9.15 shows. Figure 9.16 shows that the adaptive controller is

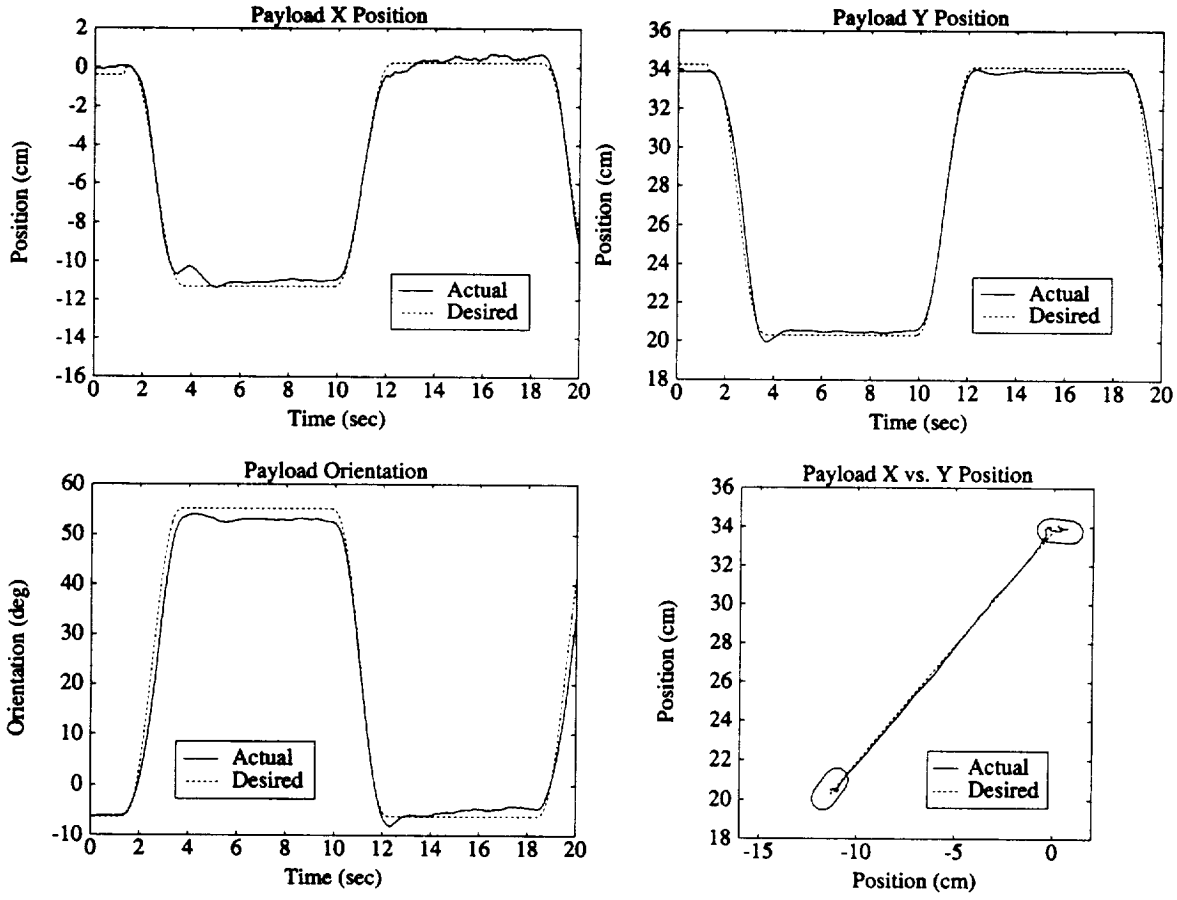


Figure 9.15: **Large-Payload Trajectories—Adaptive Control Starting with Nominal Valued for Parameter Estimates**

The actual and desired payload trajectories, measured in the "port"-fixed reference frame, show the results for the baseline adaptive controller starting with the nominal payload parameters. The orientation-tracking performance seems to improve over that of the baseline controller by decreasing the overshoot.

increasing the moment of inertia of the payload to decrease the orientation overshoot. That the moment-of-inertia estimate is monotonically increasing seems to rule out spring forces as the cause of the overshoots in the baseline controller. Spring forces will tend to aid the controller in certain

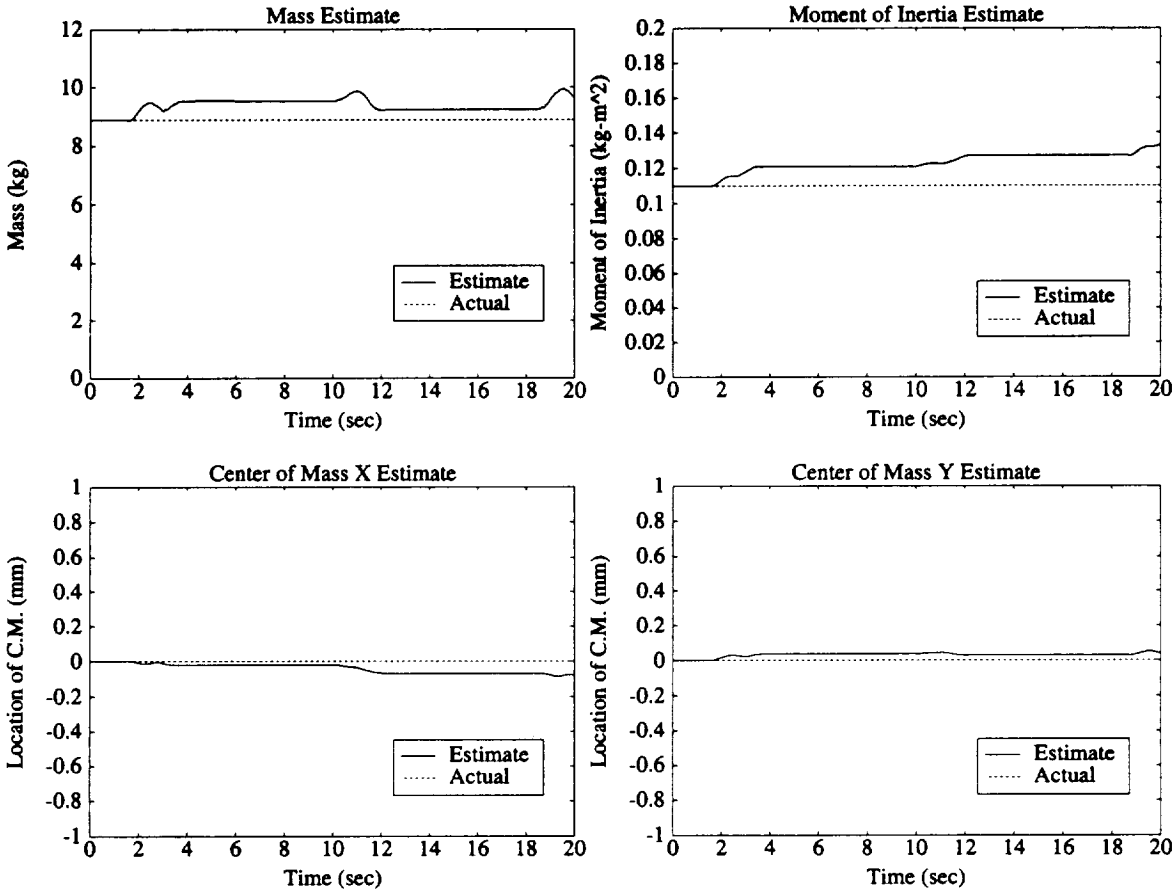


Figure 9.16: **Large-Payload Parameter Estimates—Adaptive Control Starting with Nominal Values for Parameter Estimates**

The time histories of the payload parameter estimates show that the adaptive controller is increasing the moment-of-inertia estimate to improve the orientation-tracking performance.

configurations, while hindering the controller in other configurations; this will be exhibited in the parameter estimate time-histories by increases in the estimate for one slew, then decreases for the slew in the opposite direction. This is not what the moment-of-inertia estimate in Figure 9.16 shows.

Motor saturation, therefore, is the probable cause of the overshoots, and the adaptive controller compensates for it by increasing the inertia of the payload parameters estimates to improve the trajectory-tracking performance.

9.3.3 Nonadaptive Controller with Incorrect Payload Parameters

The nonadaptive controller given the smaller payload parameters performs as poorly as the nonadaptive controller controlling the small payload using the larger payload parameters. Figure 9.17 shows the time-histories of the actual and desired payload position. This control is also unacceptable.

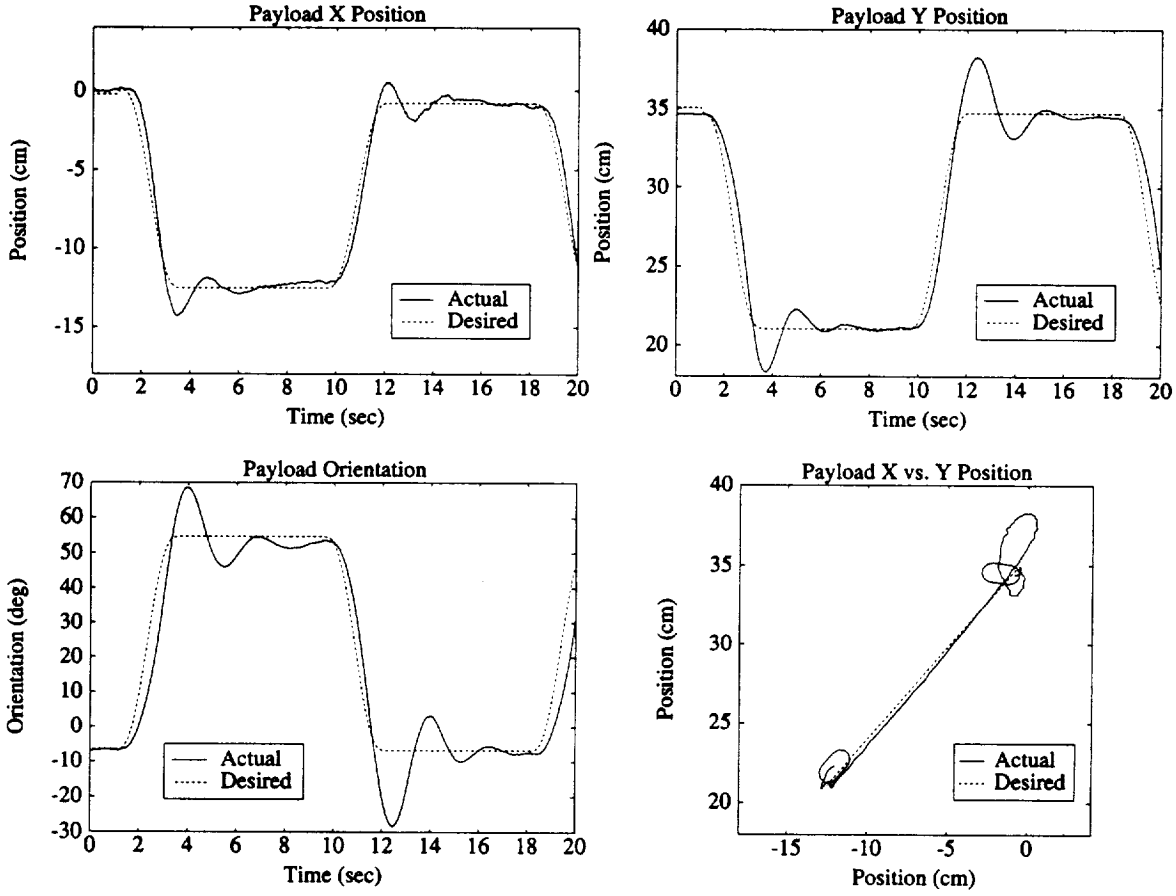


Figure 9.17: **Large-Payload Trajectories—Nonadaptive Control Starting with Incorrect Values for Parameter Estimates**

The actual and desired payload trajectories, measured in the "port"-fixed reference frame, show the results for the baseline nonadaptive *controller actually controlling the larger payload, but using the smaller-payload values for its parameters.*

9.3.4 Adaptive Controller with Incorrect Payload Parameters

Enabling the adaptive control improves controller performance, as Figure 9.18 shows. The improve-

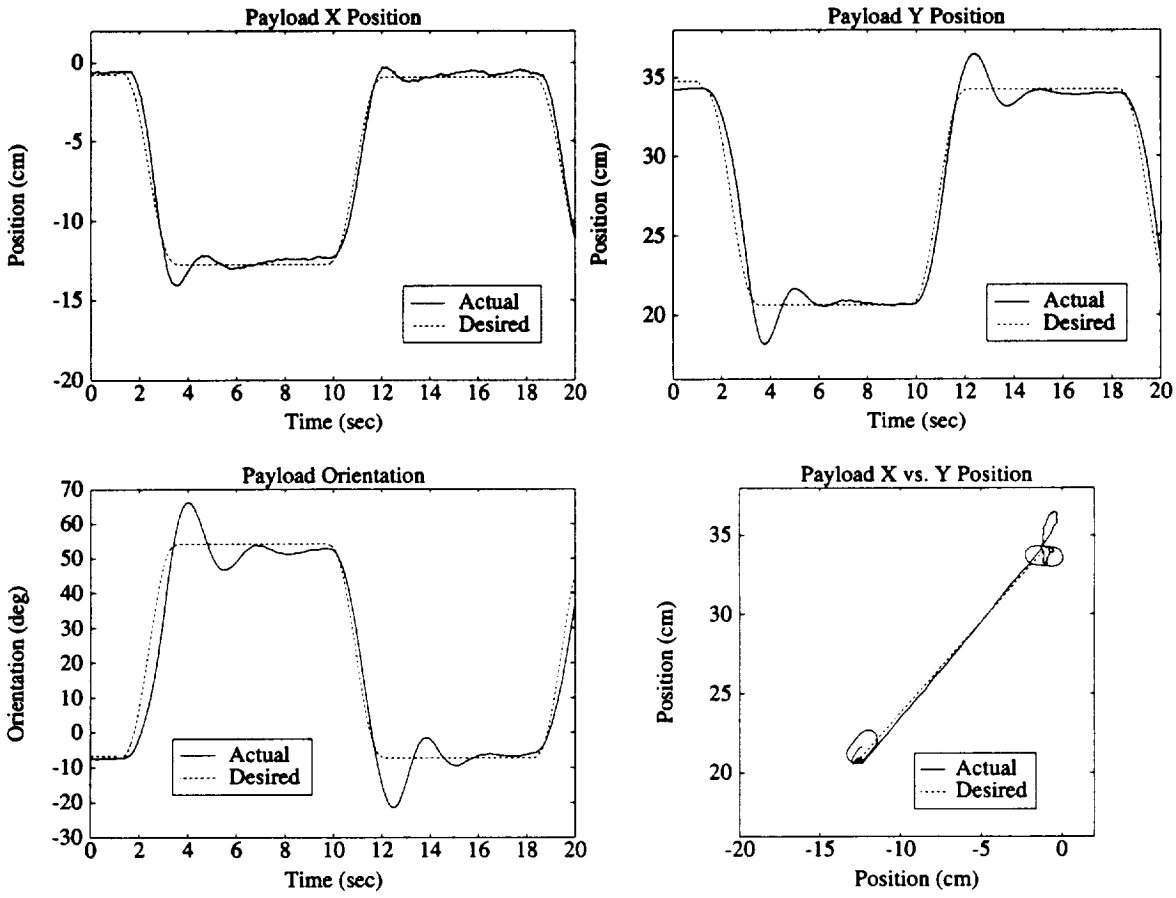


Figure 9.18: **Large-Payload Trajectories—Adaptive Control Starting with Incorrect Values for Parameter Estimates**

The actual and desired payload trajectories, measured in the "port"-fixed reference frame, show the results for the adaptive *controller controlling the large payload, but starting with the smaller-payload values for its parameters. The plots show that although the performance improves over that of the nonadaptive controller, the improvement is not as rapid as that for the adaptive control for the small payload*

ment, however is not as dramatic as that of the adaptation for the smaller payload (Figures 9.9). The trajectory-tracking errors during the slews are not large enough to make the parameters converge quickly, as Figure 9.19 illustrates. These time-histories show that the parameters have not yet converged after three slews. Higher adaptive update gains will improve the adaptation rate, but may deteriorate the performance when controlling the smaller payload; for example, a mass estimate change of 1kg represent 10 percent of the larger payload mass, but represents 100 percent of the smaller payload mass. Thus a

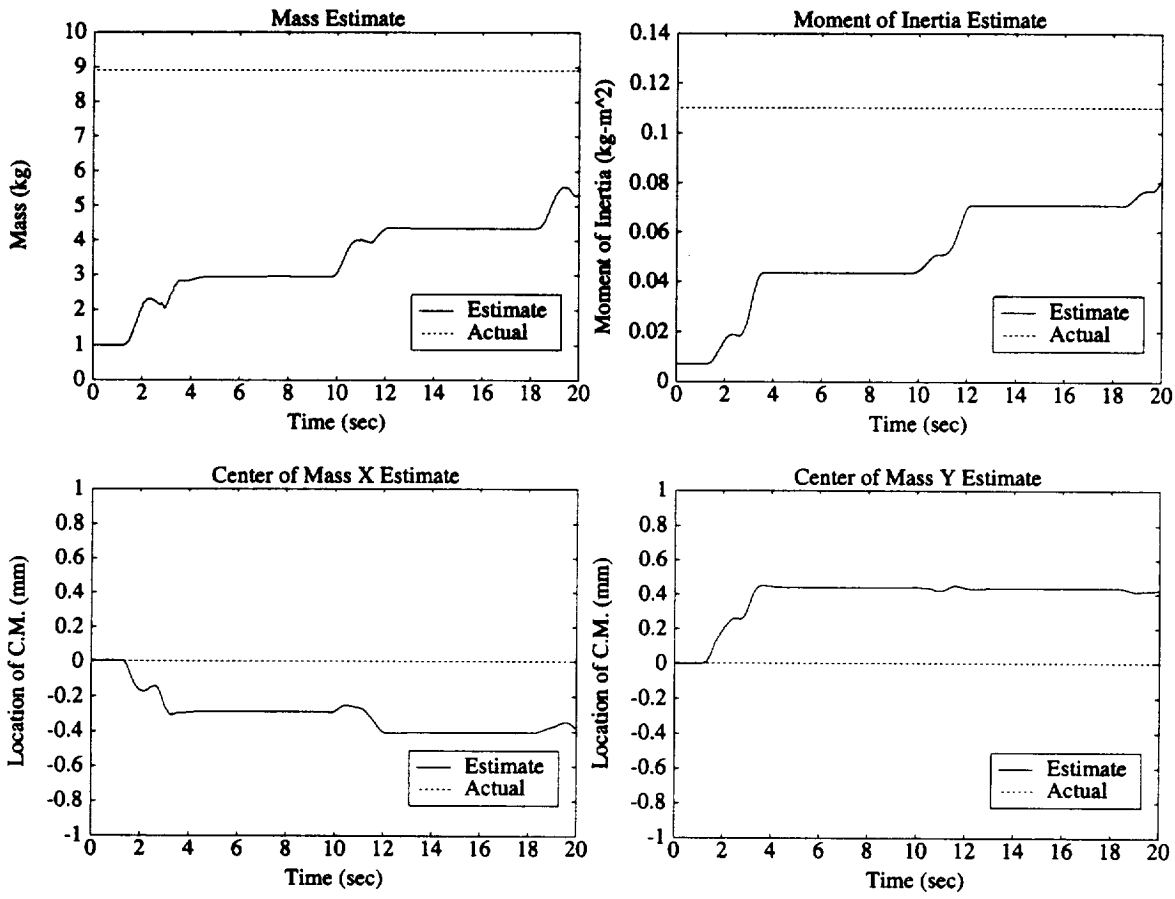


Figure 9.19: **Large-Payload Parameter Estimates—Adaptive Control Starting with Incorrect Values for Parameter Estimates**

These plots show the time histories of the parameter estimates when they are initially set to the parameters of the smaller payload.

more rapid adaptation rate for the large payload may be too high for the small payload.

Figure 9.20 shows the trajectory-tracking performance after the parameters have essentially converged. The plot shows that it is *better* than that of the baseline nonadaptive controller (see Figure 9.12). Figure 9.21 presents the converged set of payload parameter estimates. They show that the estimated moment of inertia converges to higher values than the nominal values. The adaptive controller, therefore, can improve the performance over that of the baseline controller by adjusting the parameters away from their nominal values. The adaptive controller may be compensating for some unmodelled effects to improve the trajectory-tracking errors.

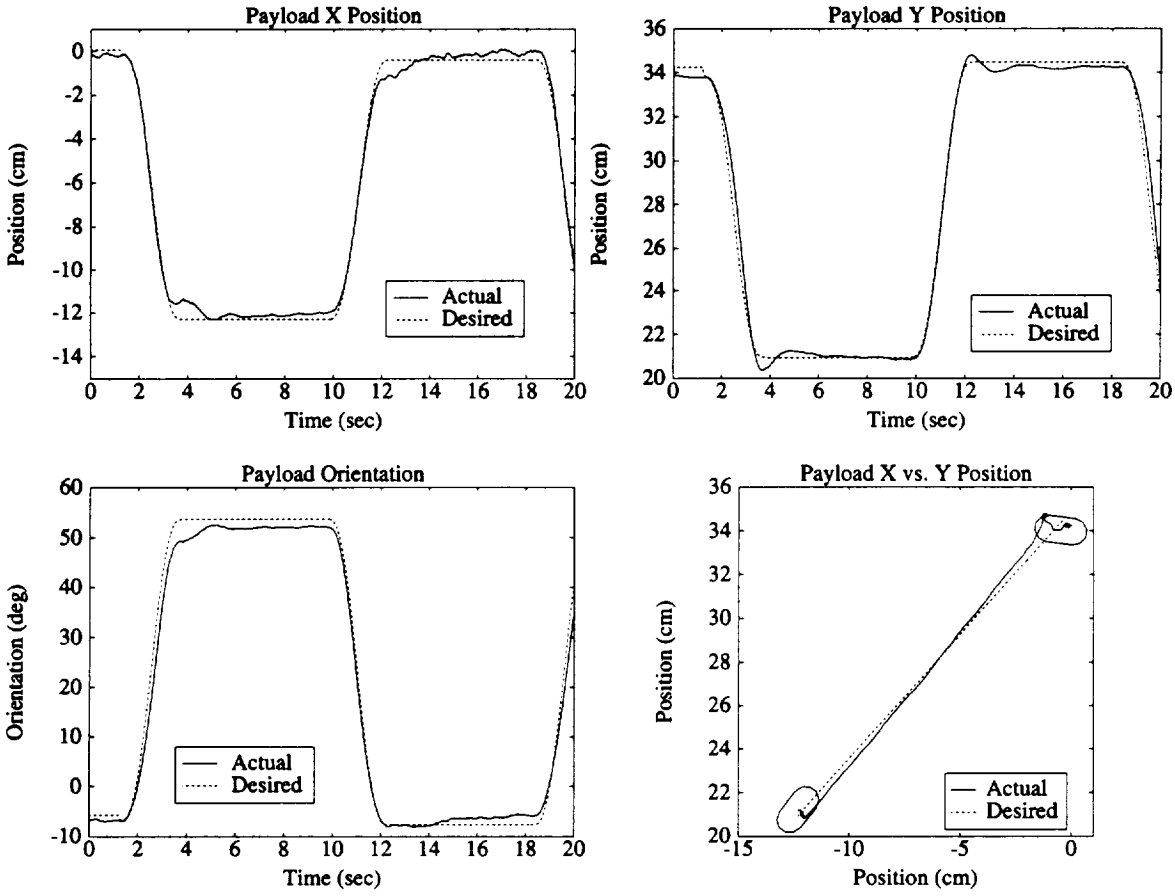


Figure 9.20: **Large-Payload Trajectories—Adaptive Control with Converged Parameter Estimates**

The actual and desired payload trajectories, measured in the "port"-fixed reference frame, show the results for the adaptive *controller after the payload parameter estimates have converged. The x and y trajectory tracking performance is similar to the baseline nonadaptive controller (Figure 9.12), but the orientation tracking is better, showing less overshoot.*

9.3.5 Large-Payload Control Summary

Figure 9.22 compares the trajectory errors of the adaptive and nonadaptive controllers starting with the incorrect payload parameters. The time histories show that the adaptive controller performance is gradually improving over that of the nonadaptive controller.

Figure 9.23 summarizes the overall performance for control of the large payload. It shows the trajectory-tracking errors of the baseline controller, the adaptive controller with converged parameters, and the nonadaptive controller utilizing the incorrect parameters. These plots demonstrate that the

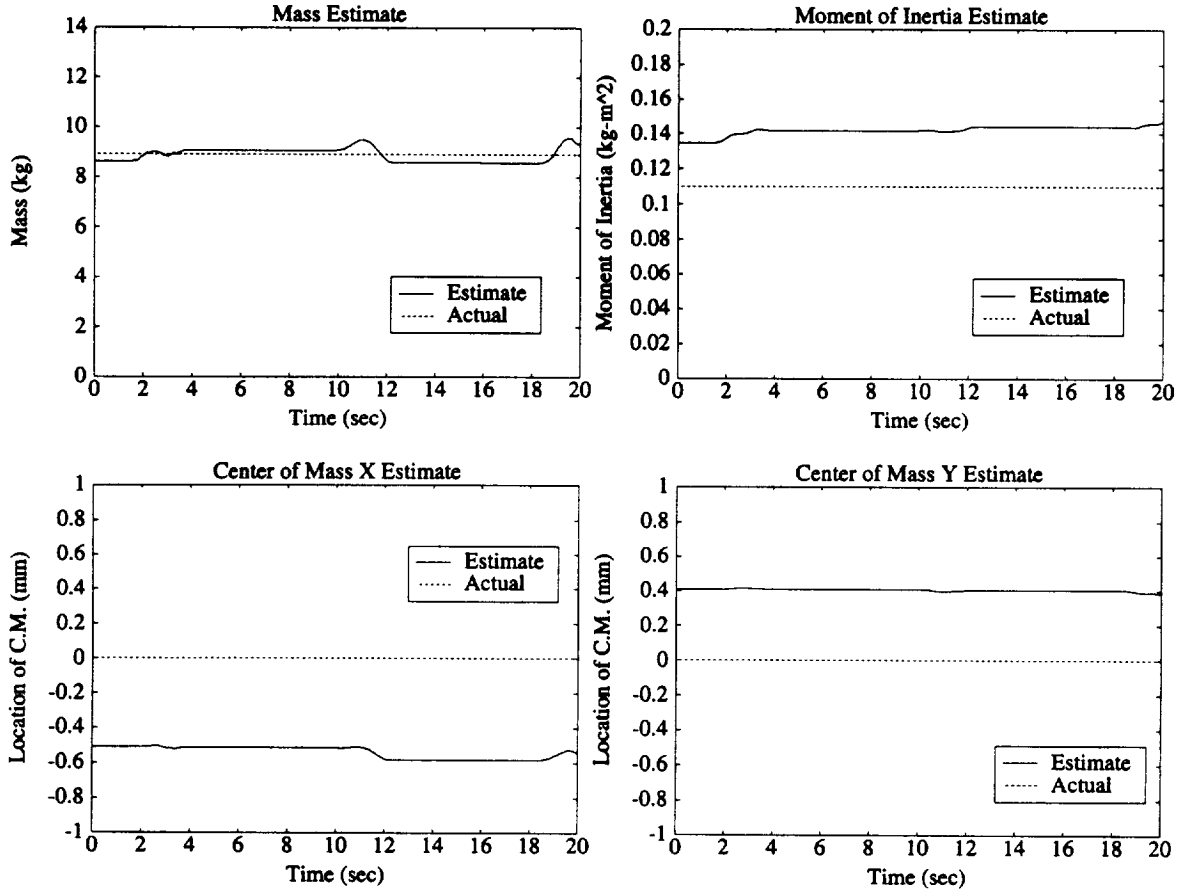


Figure 9.21: **Large-Payload Parameter Estimates—Adaptive Control with Converged Parameter Estimates**

These plots show the time histories of the parameter estimates after they have essentially converged.

adaptive controller performed better than the baseline controller, after the parameters have converged. The nonadaptive controller using the small payload parameters is unacceptable for controlling the large payload.

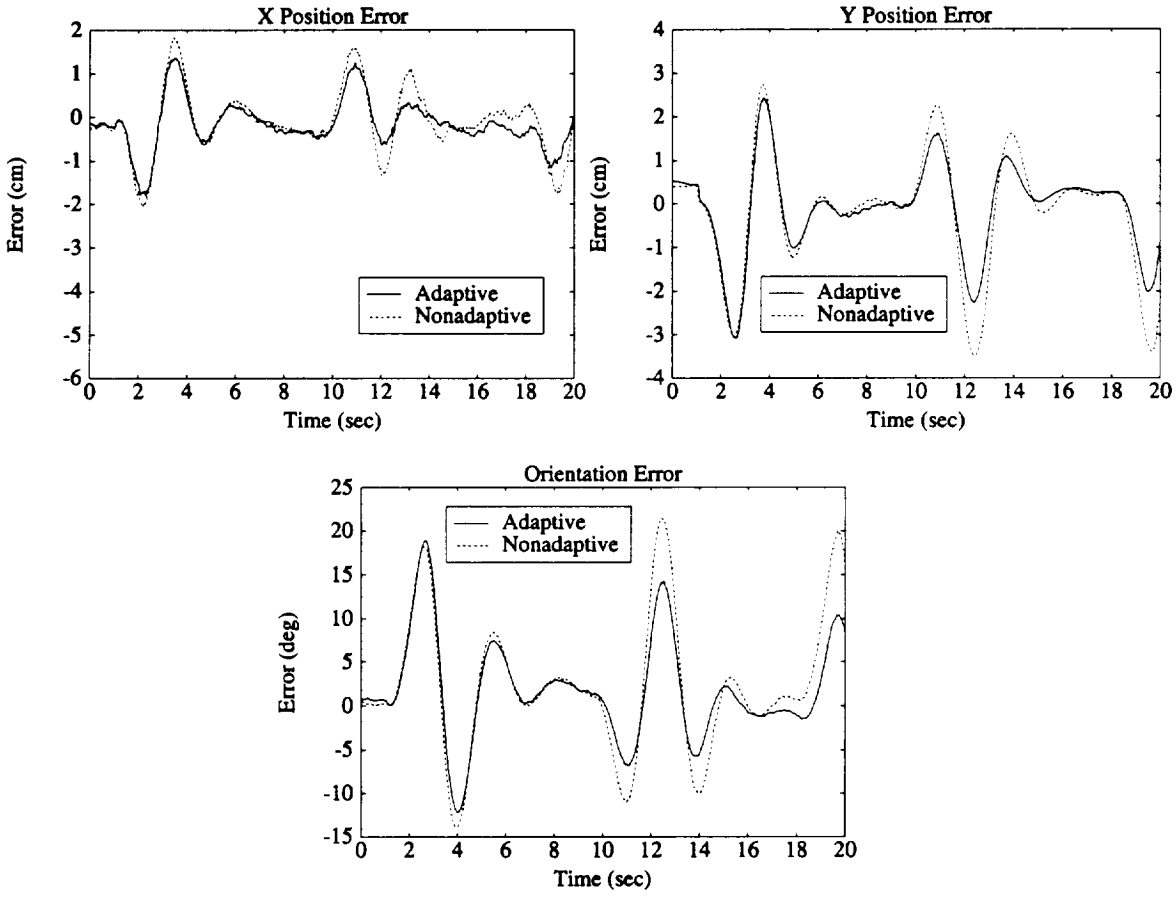


Figure 9.22: **Large-Payload Trajectory-Tracking Errors—Adaptive vs. Nonadaptive**

These plots show the time histories of the trajectory-tracking errors of adaptive and nonadaptive controllers starting with the incorrect payload parameter estimates. The adaptive controller shows gradual performance improvement over the nonadaptive controller.

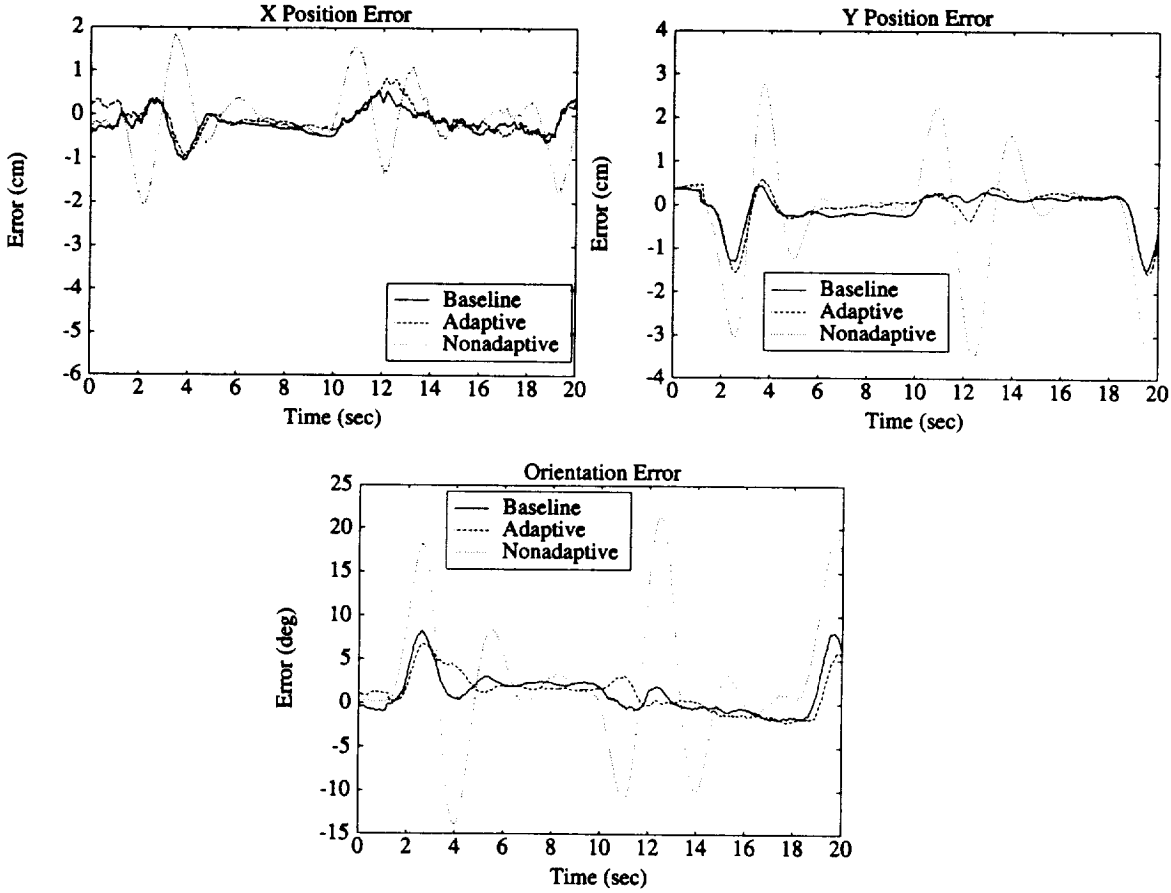


Figure 9.23: **Large-Payload Trajectory-Tracking Errors—Overall Performances**

These plots show the time histories of the trajectory-tracking errors of three controllers while controlling the large payload. The adaptive controller—after the parameters have converged—performs slightly better than the baseline nonadaptive controller using the nominal payload parameters. The performance of the nonadaptive controller using the parameters for the smaller payload is unacceptable.

9.4 Adaptation to Manipulator Parameters

This section presents results of adaptation to *manipulator* parameters utilizing the same *task*-space adaptive control framework. The results show the effectiveness of the adaptive controller in reducing trajectory-tracking errors. Endpoint control is utilized for this experiment. The baseline controller is presented first with nominal arm parameters. The inertial parameters of the right arm are then set to zero and results of endpoint slews are shown without and with adaptation.

9.4.1 Nonadaptive Controller with Nominal Arm Parameters

Figure 9.24 shows the endpoint trajectories of the baseline nonadaptive control using the nominal arm parameters. The steady-state offset are once again caused by the spring forces from wiring inside the manipulator. Integral control is disabled to show the performance of the baseline *task*-space controller[4].

9.4.2 Nonadaptive Controller with Incorrect Arm Parameters

Setting to zero the estimates for the inertial parameters of the right arm—m_1, m_2, I_1, and I_2—effectively disables the inverse-dynamics feed-forward portion of the *task*-space controller. Figure 9.25 shows the controller performance. The trajectory-following in the X direction is quite good, but there is noticeable overshoot in the Y direction. The bottom "X vs. Y" plot shows more clearly the path of the endpoint and the deterioration in performance.

[4]Integral control is enabled during actual operation to enable the capture of the free-flying object.

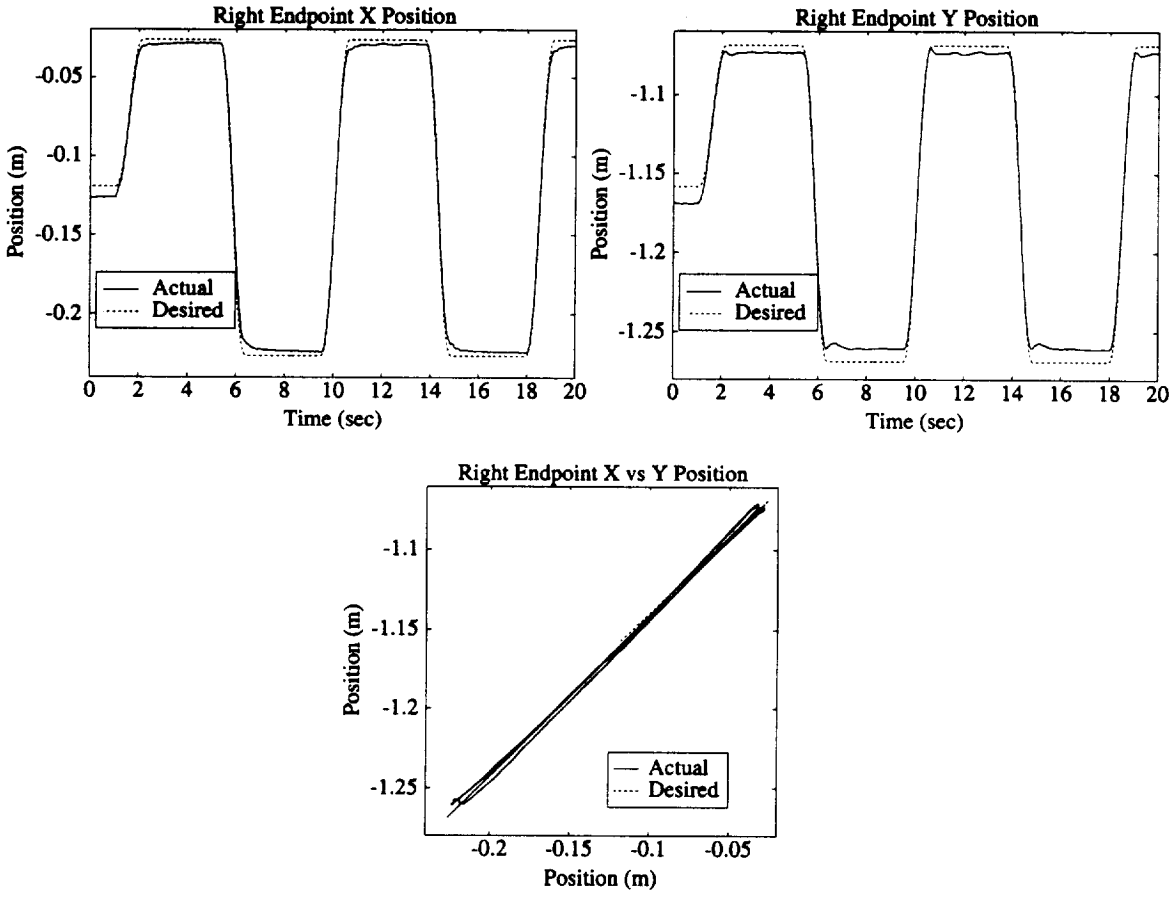


Figure 9.24: **Endpoint Position—Baseline Nonadaptive Control**

These plots show the actual and desired endpoint trajectories in inertial space of the baseline nonadaptive *controller using nominal arm parameters. The bottom plot shows the X vs. Y plot, representing an "overhead" view of the path traced out by the endpoint in inertial space.*

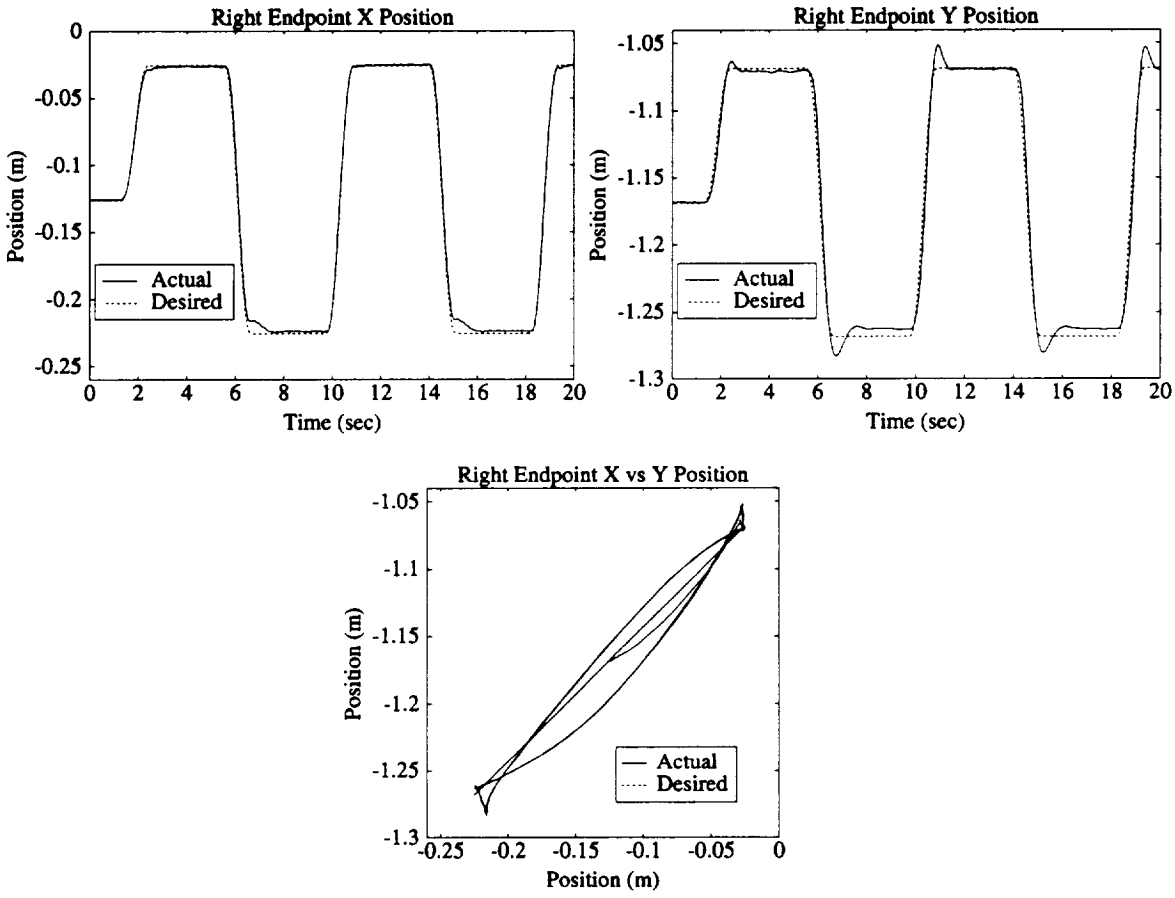


Figure 9.25: **Endpoint Position—Nonadaptive Control Starting with Incorrect Values for Parameter Estimates**

These plots show the actual and desired endpoint trajectories in inertial space of the nonadaptive *controller using zero as estimates for the arm inertial parameters.*

9.4.3 Adaptive Controller with Incorrect Arm Parameters

Enabling adaptive control improves the controller performance, as Figure 9.26 illustrates. Comparing with the baseline controller performance of Figure 9.24 shows that adaptive control performance is as good as the baseline controller by the third slew. Figure 9.27, however, shows that the parameters

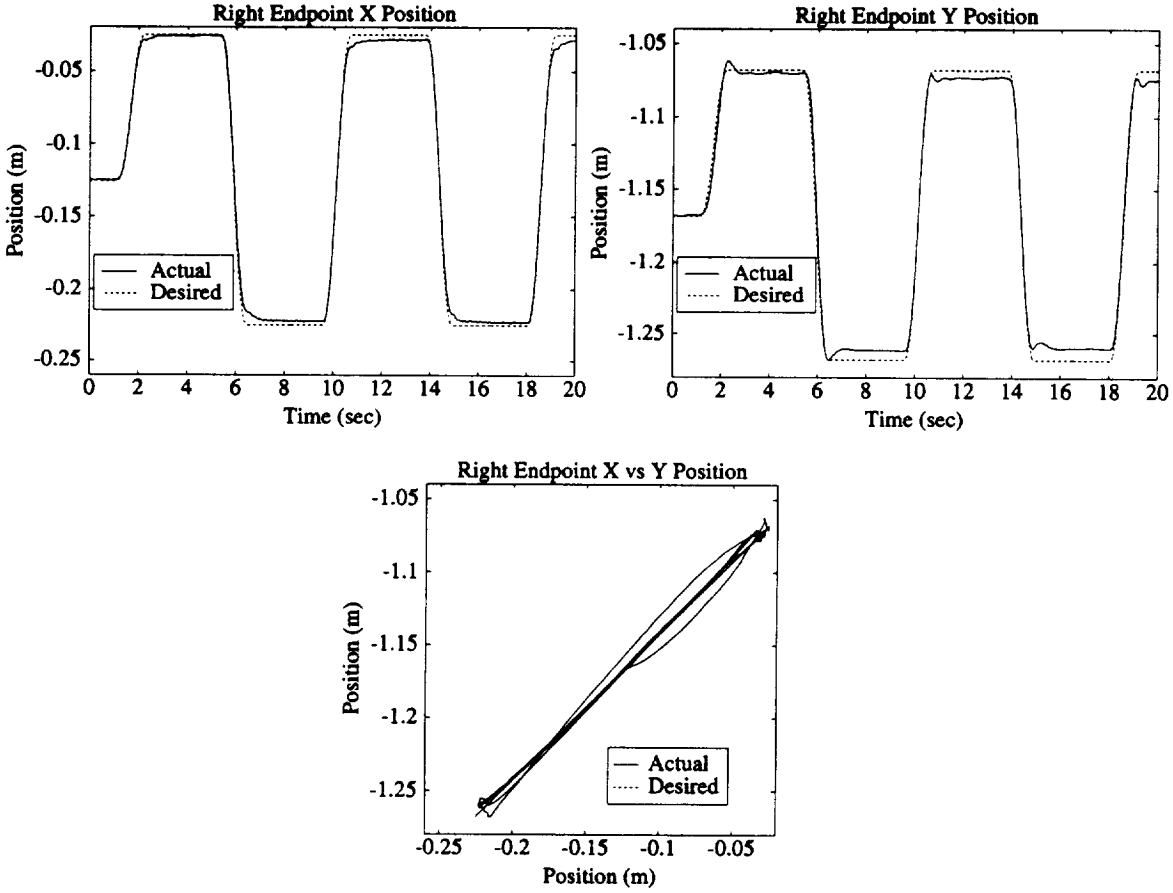


Figure 9.26: Endpoint Position—Adaptive Control Starting with Incorrect Values for Parameter Estimates

These plots show the actual and desired endpoint trajectories in inertial space of the adaptive *controller starting with zero as estimates for the arm inertial parameters.*

did not converge toward their nominal values. The PD controller did reasonably well in tracking the back-and-forth endpoint slews, so there was not enough tracking errors for the adaptive controller to allow good parameter convergence. Endpoint motions with more excitation is necessary to make the parameters converge. Nevertheless, the *task*-space adaptive controller provides good trajectory-tracking

performance.

9.5 Summary

The new *task*-space adaptive controller can provide both payload adaptation and manipulator adaptation to improve trajectory following. For the payload, where there are few parameters, the simple trajectories provide sufficient excitation to allow identification of the payload parameters to converge close to their correct values. In addition, an "incorrect" set of parameters can actually improve the trajectory-tracking performance by compensating for unmodelled effects[5]. The adaptive controller also improves endpoint-tracking errors; but there is not enough excitation to make identification of all the parameters containing the arm parameters converge.

[5]Uhlik [42] demonstrated a different aspect of this by showing that the use of an "incorrect" set of parameters for the robot manipulator can improve the *identification* of the payload; but that these "incorrect" manipulator parameters cannot, however, be used for stable high-performance control.

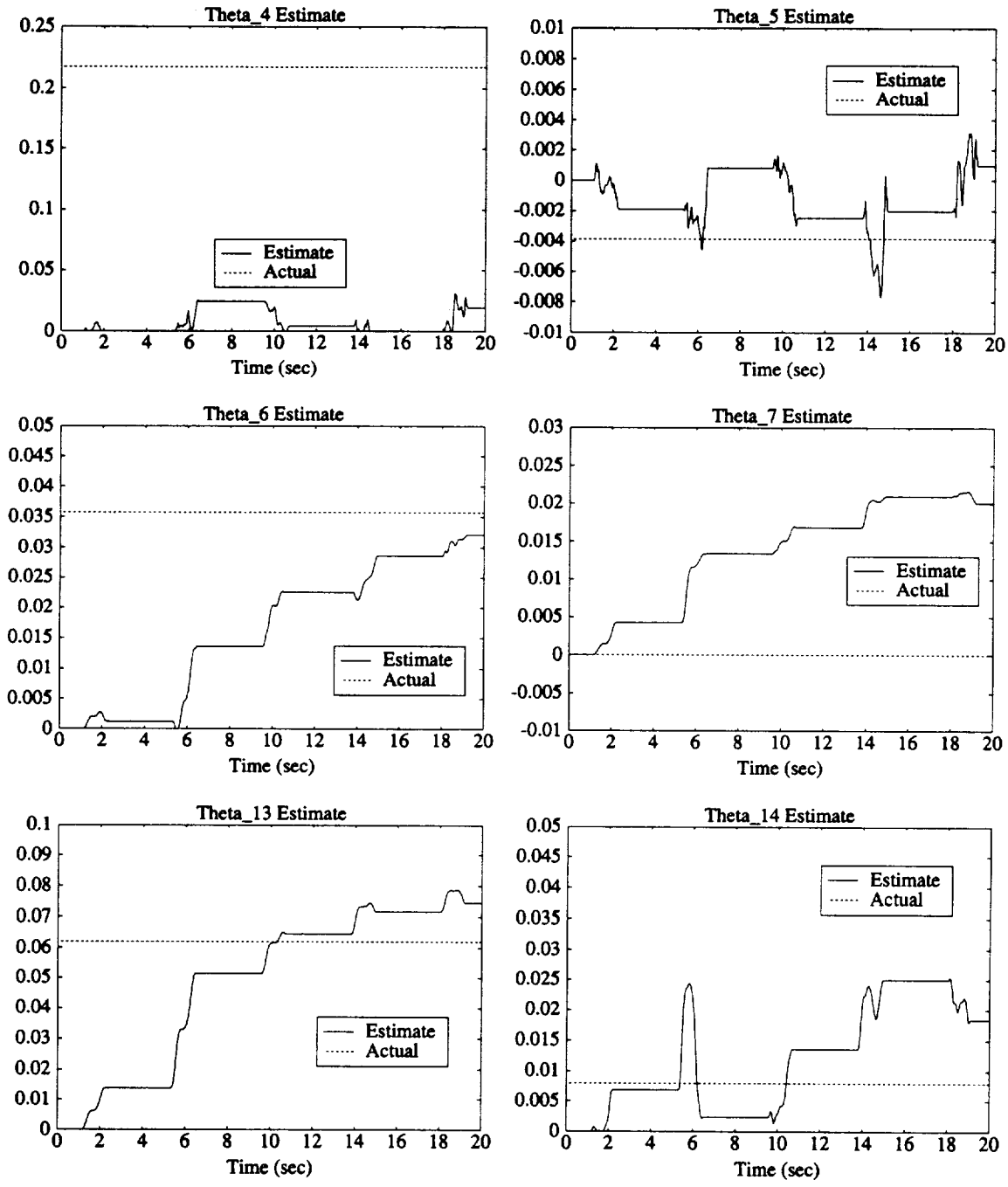


Figure 9.27: **Parameter Estimates—Adaptive Control Starting with Incorrect Values for Parameter Estimates**

These plots show the time histories of the parameters for the adaptive controller when m_1, m_2, I_1, and I_2 are initially set to zero. There is obviously not enough excitation in the trajectory to effect parameter convergence.

Chapter 10

Conclusions

This chapter summarizes the results of this research and draws some conclusions. It also describes some continuing research, and suggests possible future extensions.

10.1 Summary

This dissertation constitutes the theoretical development of a new general adaptive control framework and the experimental demonstration of the effectiveness of the new concept. This research showed that the resulting new adaptive controller maintains good stability, without sacrificing performance, in the presence of unknown or changing parameters.

The new *task*-space adaptive control framework affords the free-flying space robot—a complex system containing multiple, interacting manipulators—effective adaptive control in all control modes. The adaptive control readily handles joint-level control in the *same* manner as cooperative object control, needing only the appropriate Jacobian and torque mapping for the control mode.

The *task*-space adaptive control is a general algorithm for systems with rigid members. The theoretical development does not restrict the system to be just a free-flying space robot with two manipulators operating in two dimensions: A system of any number of fixed or free-flying robots with any number of rigid, nonredundant manipulators in three-dimensions may take advantage of this adaptive controller.

The *system concatenation* concept for control yields efficient, incremental generation of system models for multiple, interacting systems. New manipulators added to the system are typically represented in

the system models as additional block-diagonal matrices. The models existing before the addition are unaffected. The block-diagonal system matrices also lend themselves well to parallel computing.

System concatenation also keeps the adaptable parameters of each subsystem separated when incorporated into the *task*-space adaptive algorithm. This allows operators to maximize their intuition and maximize the utility of *a priori* knowledge of the controlled system. Applying heavier adaptation weighting on the parameters of a subsystem that are poorly known ensures a quicker convergence of the adaptive algorithm. The separable parameters furnish an elegant method of weighting the parameter subspaces for faster convergence without the need for swapping adaptive controllers.

The modularity of the *task*-space adaptive controller means easy implementation and multiple-control-mode support with minimal effort. The basic controller blocks and adaptive update modules do not change when switching control modes. Only the Jacobian and torque mapping modules, which typically involve easily derived kinematic relationships, need to change. Since the vector of adaptable parameters also does not change with control modes, there are no "glitches" during control mode switches.

The experimental results show that the new adaptive control achieves robustness toward plant changes with little cost to performance. In fact, the trajectory-following plots indicate that the adaptive controller performs *better* than the nominal controller, illustrating that the new adaptive control can compensate strongly for the effects of mismodelled and unmodelled parts of the system[1].

The *system concatenation* and *task*-space concepts, even without adaptation, comprise an elegant formulation of control of complex systems. They formulate multiple-manipulator control as a complete system in a more "traditional" manner. This allows the well-known computed-torque or inverse-dynamics analyses to be performed directly, without separating object control from manipulator control into two distinct steps.

Perhaps the most significant contribution of the *task*-space adaptive control framework is that it *is* a framework. A particular joint-space adaptive controller was extended to provide the adaptive portion of the framework; but the framework does not require that particular adaptive scheme. Other adaptive controllers initially designed for a single-manipulator robot may take advantage of this framework to be extended to multiple, cooperative-manipulator control.

[1]The laboratory's vision systems, which the author built early on, were essential to the success of these experiments.

10.2 Continuing Research

Ongoing research at the Aerospace Robotics Laboratory (ARL) and Information Sciences Laboratory (ISL) of Stanford University naturally complements the work presented in this thesis. One area is the use of recursive algorithms to achieve even more efficient controller implementations. Another is in the control of multiple, cooperating manipulators possessing kinematic redundancy.

The *task*-space adaptive control framework makes extensive use of Jacobian-like matrices to effect control in the *task* space. The algorithm requires both the Jacobian and its matrix inverse. As the system complexity grows, these computations may become prohibitively expensive. Recursive implementation of the inverse Jacobian have shown that computations can be greatly reduced even for a moderately complex system.

Additionally, the *system concatenation* concept can be carried to the limit, where each link of a manipulator represents a subsystem. At this extreme, the inverse dynamics control can also be implemented recursively. Coupled with recursive Jacobian implementation, these order-N algorithms provide very computationally efficient controllers. Research is progressing on extending these algorithms to the control of multiple cooperating-manipulator robots.

A redundant manipulator possesses more degrees of freedom than are needed for definitive control. Additional degrees of freedom afford the manipulator the ability to maneuver around obstacles, to save fuel, and to avoid kinematic singularity, without affecting the primary control objectives. As one example of its usefulness, this ability allows a manipulator to work in a cluttered environment with relative ease. The challenge in controlling a redundant manipulator involves developing an endpoint controller that will "naturally" avoid obstacles and singularities without operator intervention. Many researchers have already developed redundant controllers for single manipulators. Ongoing research at ARL is planned to formulate control for multiple, cooperating, redundant manipulators, but to do so in a manner that is easily extensible as more manipulators are added.

ARL is also collaborating with the Stanford Computer Science Robotics Laboratory (CSRL) to merge sophisticated path-planning algorithms with real-world robotic systems. The initial experiments are performed utilizing fixed-base robots, and the algorithms will be transferred to the Multiple-Manipulator Space Robot facility in the near future.

10.3 Suggestions for Future Research

An obvious area of further research is to extend both the recursive algorithms and the multiple redundant-manipulator control work to the realm of adaptive control. The combination of *task*-space adaptive control, recursive algorithms, and redundant-manipulator control capabilities will form a very powerful and useful control framework.

The *task*-space adaptive control has been developed for manipulator systems composed of connected rigid bodies. The applicability to systems possessing flexibility must be examined. Two types of flexibility can exist in robots: drive flexibility and flexibility distributed along the links of a manipulator. The former is quite common in industrial robots, and both are important in the Space Shuttle Remote Manipulator System (RMS); the latter is expected to be a concern in space-based manipulators where increased link flexibility is traded off for weight reduction. Since flexibility can severely limit manipulator performance, controlling well when it is present is important. Major pioneering experimental research has already been completed at ARL on the quick precise control of very flexible manipulator arms and on manipulators with flexible joint-drive systems [36, 31, 42].

Although the *task*-space adaptive control does not yet expressly address distributed flexibility, its applicability is high for robots with joint flexibility. The successful application already completed of local joint-torque control in cooperating manipulators with joint flexibility makes one quite optimistic about the direct application of *task*-space adaptive control: The joint-torque inner-loop control makes the actuators behave as perfect torque sources, hiding the flexibility from the adaptive controller. Experimentation, of course, must be conducted to determine whether further extensions must be made to handle joint flexibility. Still further research is needed to determine the applicability of the *task*-space adaptive control framework for distributed flexibility.

This dissertation demonstrates the *task*-space adaptive control framework with a specific adaptive update algorithm. But since this framework does not limit the choice of the adaptive controller, further valuable study can be made to compare and contrast the performance of other adaptive schemes, including exponentially forgetting least squares (EFLS) algorithms that take advantage of past data histories. To illustrate this, the current implementation has difficulties identifying the spring forces caused by tubing and wiring in our laboratory space-robot manipulators. The EFLS algorithms may be able to improve the parameter convergence for such terms.

Although the experimental results presented in this thesis indicate that the adaptive controller is

capable of improving the performance by overcoming effects of unmodelled dynamics, further study needs to be done to classify the explicit classes of unmodelled effects in which the *task*-space adaptive control remains effective.

The *task*-space adaptive controller can effectively adapt to an unknown payload, but only after the robot has acquired it. The capture algorithm in the current hierarchical controller plots a straight intercept trajectory, based on the object's position and velocity. This strategy is valid for objects significantly less massive than the robot, but may be fatal for very massive objects. Since the adaptive controller will not get a chance to determine which is the case before the capture, more sophisticated capture algorithms need to be developed. The intercept trajectory, for example, should parallel that of the object—matching both position and velocity—to minimize the danger while capturing the object.

Appendix A

Supporting Calculations for Lyapunov Proof

A.1 Proof of Identities for $\mathbf{M}_D$

The definition of $\mathbf{M}_D$ is repeated here:

$$\mathbf{M}_D(\mathbf{q},\mathbf{y}) \triangleq \sum_{i=1}^{n} \frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\mathbf{y}\mathbf{e}_i^T \tag{A.1}$$

where $\mathbf{e}_i \in \mathbb{R}^n$ is the ith unit vector.

Identity 1:

$$\frac{\partial\left(\mathbf{M}(\mathbf{q})\mathbf{y}\right)}{\partial \mathbf{q}} = \mathbf{M}_D(\mathbf{q},\mathbf{y}) \tag{A.2}$$

Proof:

$$\begin{aligned}\frac{\partial\left(\mathbf{M}(\mathbf{q})\mathbf{y}\right)}{\partial \mathbf{q}} &= \sum_{i=1}^{n} \frac{\partial \mathbf{M}(\mathbf{q})\mathbf{y}}{\partial q_i}\mathbf{e}_i^T \\ &= \sum_{i=1}^{n} \frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\mathbf{y}\mathbf{e}_i^T \\ &\underset{\text{(A.1)}}{=} \mathbf{M}_D(\mathbf{q},\mathbf{y})\end{aligned}$$

Identity 2:

$$\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\mathbf{y} = \mathbf{M}_D(\mathbf{q},\mathbf{y})\dot{\mathbf{q}} \tag{A.3}$$

Proof:

$$\begin{aligned}
\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}})\mathbf{y} &= \left(\sum_{i=1}^{n}\frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\dot{q}_i\right)\mathbf{y} \\
&= \left(\sum_{i=1}^{n}\frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\mathbf{y}\dot{q}_i\right) \\
&= \left(\sum_{i=1}^{n}\frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\mathbf{y}\mathbf{e}_i^T\dot{\mathbf{q}}\right) \\
&\underset{(A.1)}{=} \mathbf{M}_D(\mathbf{q},\mathbf{y})\dot{\mathbf{q}}
\end{aligned}$$

Identity 3:

$$\mathbf{M}_D(\mathbf{q},\mathbf{y})\mathbf{z} = \mathbf{M}_D(\mathbf{q},\mathbf{z})\mathbf{y} \tag{A.4}$$

Proof:

$$\begin{aligned}
\mathbf{M}_D(\mathbf{q},\mathbf{y})\mathbf{z} &\underset{(A.1)}{=} \left(\sum_{i=1}^{n}\frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\mathbf{y}\mathbf{e}_i^T\right)\mathbf{z} \\
&= \left(\sum_{i=1}^{n}\frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\mathbf{y}z_i\right) \\
&= \left(\sum_{i=1}^{n}\frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}z_i\mathbf{y}\right) \\
&= \left(\sum_{i=1}^{n}\frac{\partial \mathbf{M}(\mathbf{q})}{\partial q_i}\mathbf{z}\mathbf{e}_i^T\right)\mathbf{y} \\
&\underset{(A.1)}{=} \mathbf{M}_D(\mathbf{q},\mathbf{y})\dot{\mathbf{q}}
\end{aligned}$$

A.2 Representation for C

The representation for the matrix of Coriolis and centrifugal terms, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, in the equations of motion for a robot with rigid links is not unique, although the vector, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$, *is* uniquely specified. The stability proof found in this thesis utilizes a particular representation:

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}}) - \frac{1}{2}\mathbf{M}_D^T(\mathbf{q},\dot{\mathbf{q}}) \tag{A.5}$$

The following shows that this is a valid choice:

$$\begin{aligned}
\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} &\underset{(A.5)}{=} \left(\mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}}) - \frac{1}{2}\mathbf{M}_D^T(\mathbf{q},\dot{\mathbf{q}})\right)\dot{\mathbf{q}} \\
&\underset{(A.2,A.3)}{=} \left(\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}}) - \frac{1}{2}\left(\frac{\partial\left(\mathbf{M}(\mathbf{q})\mathbf{y}\right)}{\partial \mathbf{q}}\right)^T\right)\dot{\mathbf{q}}
\end{aligned}$$

$$\underset{(3.15)}{=} \quad \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$$

Please note, however, that:

$$\left(\mathbf{M}_D(\mathbf{q},\dot{\mathbf{q}}) - \frac{1}{2}\mathbf{M}_D^T(\mathbf{q},\dot{\mathbf{q}})\right) \neq \left(\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}}) - \frac{1}{2}\left(\frac{\partial\left(\mathbf{M}(\mathbf{q})\mathbf{y}\right)}{\partial \mathbf{q}}\right)^T\right) \tag{A.6}$$

This representation for $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ also does not satisfy the skew-symmetry property needed in the sliding-mode adaptive controller [37]:

$$\mathbf{y}^T\left(\dot{\mathbf{M}}(\mathbf{q},\dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\right)\mathbf{y} = \mathbf{0} \quad \forall\, \mathbf{y} \in \mathbb{R}^n \tag{A.7}$$

This property is not required for the Lyapunov proof in this dissertation. Additionally, note that this representation for $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, Equation (A.5), is useful only in the stability proof. It is not needed for implementation of the control or adaptive update laws, which use the combination, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$.

A.2.1 Example

To illustrate the different representations, use once again the planar two-link arm example in Figure 2.1. The choice of $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ utilized in Equation (2.5) is:

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} -m_2 l_1 l_2^* \sin(q_2)\dot{q}_2 & -m_2 l_1 l_2^* \sin(q_2)\left(\dot{q}_1 + \dot{q}_2\right) \\ m_2 l_1 l_2^* \sin(q_2)\dot{q}_1 & 0 \end{bmatrix} \tag{A.8}$$

which satisfies the skew-symmetric property.

Using the representation defined by Equation (A.5) yields:

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} 0 & -m_2 l_1 l_2^* \sin(q_2)\left(2\dot{q}_1 + \dot{q}_2\right) \\ m_2 l_1 l_2^* \sin(q_2)\left(\dot{q}_1 - \frac{1}{2}\dot{q}_2\right) & \frac{1}{2} m_2 l_1 l_2^* \sin(q_2)\dot{q}_2 \end{bmatrix} \tag{A.9}$$

Using either representation for $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, the following is true:

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} = \begin{bmatrix} -m_2 l_1 l_2^* \sin(q_2)\left(2\dot{q}_1 + \dot{q}_2\right)\dot{q}_2 \\ m_2 l_1 l_2^* \sin(q_2)\dot{q}_1^2 \end{bmatrix} \tag{A.10}$$

showing that $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$ is uniquely specified.

A.3 Control and Adaptive Law Transformations

This section shows the equivalency of the control and adaptation laws developed for equations of motion written in terms of $\dot{\mathbf{q}}$'s and those written in terms of generalized speeds.

The the control and adaptation laws derived for equations of motion in terms of generalized speeds is repeated here as:

$$\mathbf{F} \underset{(4.4)}{=} \widehat{\mathbf{M}}'(\mathbf{q})\dot{\mathbf{u}}_d + \widehat{\mathbf{C}}'(\mathbf{q},\mathbf{u}_d)\mathbf{u}_d + \widehat{\mathbf{G}}'(\mathbf{q}) + \mathbf{K}'_V\tilde{\mathbf{u}} + \mathbf{W}^{-T}\mathbf{K}_P\tilde{\mathbf{q}} \tag{A.11}$$

$$\dot{\hat{\boldsymbol{\theta}}} \underset{(4.6)}{=} \boldsymbol{\Gamma}\mathbf{Y}'^{T}(\mathbf{q},\mathbf{u}_d,\mathbf{u}_d,\dot{\mathbf{u}}_d)\left(\tilde{\mathbf{u}} + c\mathbf{W}^T\tilde{\mathbf{q}}\right) \tag{A.12}$$

where

$$\mathbf{F} \underset{(4.3)}{=} \mathbf{W}^{-T}\boldsymbol{\tau} \tag{A.13}$$

A.3.1 Control Law Equivalence

Given the transformation equations derived in Chapter 4,

$$\begin{aligned}
\mathbf{F} &= \mathbf{W}^{-T}\boldsymbol{\tau} \\
\widehat{\mathbf{M}}'(\mathbf{q}) &= \mathbf{W}^{-T}\widehat{\mathbf{M}}(\mathbf{q})\mathbf{W}^{-1} \\
\widehat{\mathbf{C}}'(\mathbf{q},\mathbf{u}) &= \mathbf{W}^{-T}\widehat{\mathbf{C}}(\mathbf{q},\mathbf{W}^{-1}\mathbf{u})\mathbf{W}^{-1} \\
\widehat{\mathbf{G}}' &= \mathbf{W}^{-T}\widehat{\mathbf{G}}(\mathbf{q}) \\
\mathbf{Y}'(\mathbf{q},\mathbf{u},\mathbf{u},\dot{\mathbf{u}})\boldsymbol{\theta} &= \mathbf{W}^{-T}\mathbf{Y}(\mathbf{q},\mathbf{W}^{-1}\mathbf{u},\mathbf{W}^{-1}\mathbf{u},\mathbf{W}^{-1}\dot{\mathbf{u}})\boldsymbol{\theta}
\end{aligned} \tag{A.14}$$

substitute them into Equation (A.11):

$$\begin{aligned}
\mathbf{W}^{-T}\boldsymbol{\tau} &= \mathbf{W}^{-T}\widehat{\mathbf{M}}(\mathbf{q})\mathbf{W}^{-1}\dot{\mathbf{u}}_d + \mathbf{W}^{-T}\widehat{\mathbf{C}}(\mathbf{q},\mathbf{W}^{-1}\mathbf{u}_d)\mathbf{W}^{-1}\mathbf{u}_d + \mathbf{W}^{-T}\widehat{\mathbf{G}}(\mathbf{q}) \\
&\quad + \mathbf{W}^{-T}\mathbf{W}^T\mathbf{K}'_V\tilde{\mathbf{u}} + \mathbf{W}^{-T}\mathbf{K}_P\tilde{\mathbf{q}} \\
&\underset{(4.1,4.8)}{=} \mathbf{W}^{-T}\left(\widehat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_d + \widehat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d + \widehat{\mathbf{G}}(\mathbf{q}) + \mathbf{W}^T\mathbf{K}'_V\mathbf{W}\dot{\tilde{\mathbf{q}}} + \mathbf{K}_P\tilde{\mathbf{q}}\right)
\end{aligned} \tag{A.15}$$

Multiply both sides of Equation (A.15) by $\mathbf{W}^T$ yields the Bayard and Wen control law,

$$\boldsymbol{\tau} \underset{(3.4)}{=} \widehat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_d + \widehat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}}_d)\dot{\mathbf{q}}_d + \widehat{\mathbf{G}}(\mathbf{q}) + \mathbf{W}^T\mathbf{K}_V\mathbf{W}\dot{\tilde{\mathbf{q}}} + \mathbf{K}_P\tilde{\mathbf{q}}$$

with a slightly different velocity gain matrix, $\mathbf{K}_V = \mathbf{W}^T\mathbf{K}'_V\mathbf{W}$.

A.3.2 Adaptation Law Equivalence

Strictly speaking, the adaptation law in Equation (A.12) is derived from the following condition:

$$0 \underset{(4.14)}{=} \tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma}^{-1} \dot{\tilde{\boldsymbol{\theta}}} + \tilde{\boldsymbol{\theta}}^T \mathbf{Y}^T(\mathbf{q}, \mathbf{u}_d, \mathbf{u}_d, \dot{\mathbf{u}}_d)\,(\tilde{\mathbf{u}} + c\mathbf{W}\tilde{\mathbf{q}}) \tag{A.16}$$

Substituting the transformation equations into Equation (A.16) gives:

$$\begin{aligned} 0 &= \tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma}^{-1} \dot{\tilde{\boldsymbol{\theta}}} + \tilde{\boldsymbol{\theta}}^T \mathbf{Y}^T(\mathbf{q}, \mathbf{W}^{-1}\mathbf{u}_d, \mathbf{W}^{-1}\mathbf{u}_d, \mathbf{W}^{-1}\dot{\mathbf{u}}_d)\mathbf{W}^{-1}\,(\tilde{\mathbf{u}} + c\mathbf{W}\tilde{\mathbf{q}}) \\ &\underset{(4.1,\,4.8)}{=} \tilde{\boldsymbol{\theta}}^T \boldsymbol{\Gamma}^{-1} \dot{\tilde{\boldsymbol{\theta}}} + \tilde{\boldsymbol{\theta}}^T \mathbf{Y}^T(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)\left(\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}\right) \\ \Rightarrow \quad \dot{\tilde{\boldsymbol{\theta}}} &= -\boldsymbol{\Gamma}\mathbf{Y}^T(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)\left(\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}\right) \\ \Rightarrow \quad \dot{\hat{\boldsymbol{\theta}}} &= \boldsymbol{\Gamma}\mathbf{Y}^T(\mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d)\left(\dot{\tilde{\mathbf{q}}} + c\tilde{\mathbf{q}}\right) \end{aligned} \tag{A.17}$$

since $\dot{\tilde{\boldsymbol{\theta}}} = -\dot{\hat{\boldsymbol{\theta}}}$. Equation (A.17) *is* the Bayard and Wen adaptive update law.

This proves that the two sets of control and adaptive update laws are equivalent.

A.4 Torques to Endpoint Forces using Virtual Work

This section shows, using virtual work arguments, the familiar expression relating endpoint forces and actuator torques:

$$\boldsymbol{\tau} = \mathbf{J}^T(\mathbf{q})\mathbf{F}^{tip} \tag{A.18}$$

The work done by the manipulator actuators as they move through a virtual displacements at each joint, $\delta\mathbf{q}$, is:

$$\delta W = \boldsymbol{\tau}^T\, \delta\mathbf{q} \tag{A.19}$$

The work delivered at the endpoint of the manipulator through a virtual displacement at the endpoint, $\delta\mathbf{x}$, is:

$$\delta W = \mathbf{F}^{tip^T}\, \delta\mathbf{x} \tag{A.20}$$

These must be equal, hence:

$$\boldsymbol{\tau}^T\, \delta\mathbf{q} \underset{(A.19,A.20)}{=} \mathbf{F}^{tip^T}\, \delta\mathbf{x} \tag{A.21}$$

As the virtual displacements go to zero,

$$\delta\mathbf{x} = \frac{\partial \mathbf{x}}{\partial \mathbf{q}}\delta\mathbf{q} \tag{A.22}$$

$$= \mathbf{J}(\mathbf{q})\delta\mathbf{q} \tag{A.23}$$

Substituting Equation (A.23) into Equation (A.21) for $\delta \mathbf{x}$ gives:

$$\boldsymbol{\tau}^T \delta \mathbf{q} \underset{(A.21,A.23)}{=} \mathbf{F}^{tip^T} \mathbf{J}(\mathbf{q}) \, \delta \mathbf{q} \tag{A.24}$$

Transposing both sides of Equation (A.24) gives:

$$\begin{aligned} \delta \mathbf{q}^T \boldsymbol{\tau} &\underset{(A.24)}{=} \delta \mathbf{q}^T \mathbf{J}^T(\mathbf{q}) \mathbf{F}^{tip} \\ \Rightarrow \quad \boldsymbol{\tau} &= \mathbf{J}^T(\mathbf{q}) \mathbf{F}^{tip} \end{aligned} \tag{A.25}$$

Thus, showing the relationship in Equation (A.18).

Appendix B

Point Grabber II Vision System

This appendix includes the User's Manual for the Point Grabber II Vision system, resolution-testing plots, the schematics, and the PALASM listings of the logic for the PALs (Programmable Array Logic) used in the Point Grabber II board.

B.1 User's Manual

The user's manual start on the following page.

PointGrabber II User's Manual

Introduction

Point Grabber II is a specialized single-board VMEbus-based vision processing unit that provides real-time systems with high speed, high resolution vision information from CCD cameras. It is ideally suited for applications in a robotics and control environment where high speed sampling is essential to achieve high performance.

Coupled with non-interlaced CCD television cameras and RTI's Visionserver software, Point Grabber II can provide frame updates at 60 Hz with resolutions better than 1/40 of a pixel. Over a field of view of 2 meters square, that translates to a resolution of 0.2 mm square.

To decrease the computational burden on host computers, Point Grabber II locates, digitizes, and stores only bright points in the field of view. Bright markers, such as LEDs, placed on objects of interest allows appropriate software to determine the position and orientation of the objects from Point Grabber II data. Unlike "Frame Grabbers", Point Grabber II provides data that all correspond directly to markers, eliminating the need for host computers to perform sophisticated scene analysis computations. Additionally, each Point Grabber II board can process data from two cameras simultaneously.

System Requirements

- VMEbus system
- Host CPU card
- Non-interlaced CCD camera capable of accepting external horizontal and vertical synchronization signals

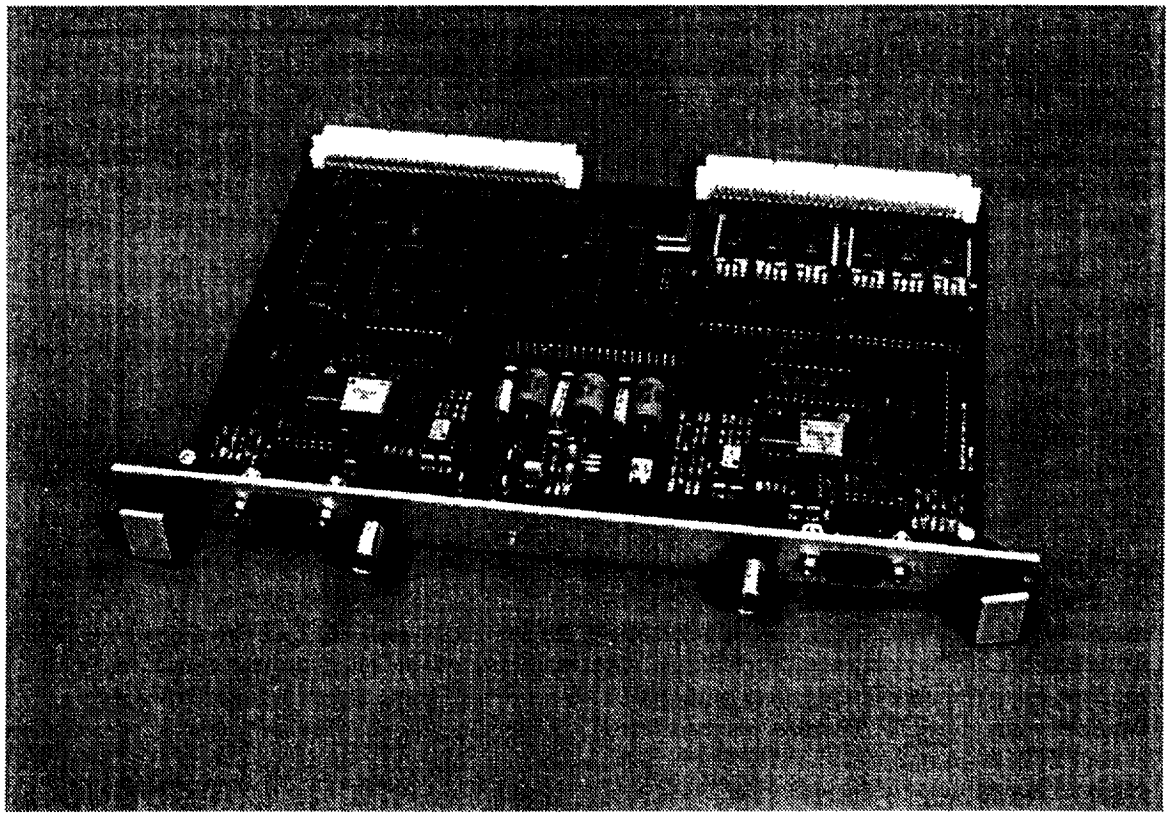


Figure B.1: **Point Grabber II Vision Board**

General Description

Point Grabber II detects, registers, and digitizes brightly illuminated targets from video signals generated by up to two cameras. Either active infrared LEDs or passive high reflectance stickers reflecting light from an incandescent source can provide the bright targets . Visible-cut, infrared-pass filters placed over the camera lenses will reject ambient visible light . Independent programmable threshold voltages for each camera video input determine the transition between bright and dark. The vision board "detects" a bright target when the video input signal from either camera rises above its respective threshold voltage.

Point Grabber II tracks the current horizontal (X) position and vertical (Y) position of the video scan of each camera. The video board can, therefore, register the position, measured in pixels, of the each bright image. The X position is measured starting from the left-hand side of the field of view, and the Y position is measured starting from the top of the field of view. By sending horizontal and vertical synchronization signals (HSYNC and VSYNC, respectively) to the CCD cameras, and by utilizing the phase-locked loops (PLL) in the cameras, Point Grabber II provides very stable tracking of the video scan. This method also ensures that, if two cameras are used, the video scans of both cameras are in strict synchronization.

Besides registering the position of bright pixels, 8-bit analog to digital (A/D) converters on the vision board digitize the video signals corresponding to those pixels. The board stores the digitized values with the position information. Vision processing software may use the digitized values to achieve the 1/40 pixel resolution. The dynamic range of Point Grabber II is user programmable. Point Grabber II automatically sets the lower reference voltage of each A/D converter to the corresponding threshold voltage for each camera. By programming the upper reference voltage to be the highest expected video input voltage, one can maximize sensitivity.

Point Grabber II stores the vision information in four (4) First-In-First-Out (FIFO) registers. One FIFO (X FIFO) records the X position, one (Y FIFO) records the Y position, and two (Z0 FIFO and Z1 FIFO) record the digitized values representing pixel brightness as seen by each camera. The vision board keeps all data in the FIFOs in strict synchronization. That is, for each bright pixel detected by one or both cameras, Point Grabber II writes data to all FIFOs. If only one camera detects a bright pixel, the digitized value stored for the other camera is zero. Additionally, the vision board writes an extra data set to the FIFOs at the end of each video frame. This data set, or frame marker, allows Point Grabber II to continue storing data for the next video frame while vision processing software is still reading data for the current frame. The implication of Point Grabber II's storage scheme is that the vision processing software also must perform FIFO reads in strict synchronization. That is, the FIFOs must be accessed an equal number of times. In the event that synchronization is lost, the software may reset the FIFOs via a hardware register.

Point Grabber II functions as a VMEbus interrupter. With interrupts enabled, the vision board generates interrupts at the end of every camera video frame—typically every 1/60 of a second. The interrupt level and interrupt vector are user programmable.

Additional registers on Point Grabber II provide separate camera enables, empty/full status of each FIFO, and board-level reset. Refer to the appropriate sections for more detailed descriptions of the location and usage of the registers mentioned in this section.

Installation

Address Jumpers

Point Grabber II can be configured to reside in any 256-byte segment in short I/O space, starting at hex address 0xFFFF0000 and ending at 0xFFFFFFFF. The address jumpers, J1 through J8, are located near the P1 connector. The jumpers represent bits 8 through 15 of the board address. With a jumper installed, the corresponding bit is set to 0. The default configuration, with only J1 installed, makes the board appear at the 256-byte segment starting at 0xFFFFFE00.

Jumper configuration	Bit value
installed (x)	0
not installed (o)	1

Jumper	J8	J7	J6	J5	J4	J3	J2	J1
Address Bit	15	14	13	12	11	10	9	8
Default setting	o	o	o	o	o	o	o	x

Camera Synchronization Timing

No further timing configuration is required to support the Pulnix TM-440S camera.

Point Grabber II provides precise synchronization with its cameras by utilizing the phase-locked-loop (PLL) circuitry found in many CCD cameras. The PLL locks on to the user configurable horizontal and vertical synchronization signals supplied by Point Grabber II.

The camera specifications must be available before the synchronization timing can be configured. The standard crystal oscillator (U17) provided with Point Grabber II has a frequency of 14.318 MHz. This value must be twice the frequency of the pixel clock of the camera. From the camera specifications, make a note of the required number of pixel clocks per horizontal line (horizontal count) and the number horizontal lines per field (vertical count) produced in the non-interlaced mode. These numbers are essential to the proper configuration of the vision board.

Use the six 4-position dip switches, S1 through S6, to configure the camera synchronization timing. The left group of three switches (S1 through S3) represents the horizontal count, while the right group (S4 through S6) represents the vertical count. Letting the ON position represent logical 1, and letting each group of dip switches represent a 12-bit binary number, set the switches for the desired horizontal count minus one and vertical count minus one, respectively.

For example, the default settings on Point Grabber II are for the Pulnix TM-440S camera, which requires a horizontal count of 455 and a vertical count of 262. The resulting switches settings are:

Switch	S1	S2	S3	S4	S5	S6
Setting (1=ON, 0=OFF)	0001	1100	0110	0001	0000	0101
Decimal value	454			261		

Register Descriptions

The Point Grabber II address space occupies a 256-byte segment selected by the address jumpers (see the Installation section). The default segment starts at hex address 0xFFFFFE00. The following register descriptions will give the addresses of the registers in terms of an offset into the selected 256-byte segment. Addresses not mentioned are unused.

FIFO Registers

Point Grabber II stores its vision image information in four (4) First-In-First-Out (FIFO) registers. Each FIFO is 9 bits wide and 512 deep. To the VMEbus, these FIFOs are read-only, 16-bit word registers.

The vision board stores the X and Y location of each bright pixel in the X and Y FIFOs, respectively. It stores the digitized value of a bright pixel from the first camera (Camera 0) in the Z0 FIFO, and that of the second camera (Camera 1) in the Z1 FIFO. Since all the FIFOs are written when a bright pixel is detected in either or both cameras, the lower 8 bits of the Z0 and Z1 FIFOs will both be nonzero only if both cameras detect bright targets simultaneously. If only one camera detects a bright target, the data in the other Z FIFO will be zero. Any vision processing software must perform an equal number of reads from each FIFO to ensure that the data from all FIFOs on each set of reads refer to the same event.

At the end of each video frame, Point Grabber II additionally stores a data set, called the frame marker, into the FIFOs. By monitoring the FIFO data for the frame marker, vision processing software can easily determine the end of a video frame, while allowing the vision board to continue storing vision data for the next video frame. This scheme takes advantage of the "double-buffering" capability of the FIFOs, and ensures that no vision information is lost while performing vision processing.

Because the FIFOs have finite depth, it is possible for them to become full. This will happen if too many bright targets appear in the field of view, which may be caused by excessive illumination or inappropriate threshold voltages. Similarly, it is possible for the FIFOs to be empty, which indicates that no data is currently available. When used as a bus interrupter, empty FIFOs also indicate an error, since they should contain at least a frame marker data set. Either of these conditions indicate error, and vision processing software should check the status register (see below) before proceeding to read FIFO data. The recommended action for empty/full error is to reset the FIFOs (see below).

X FIFO (R) (0xD0)

Bits 15-9	Bits 8-0
Unused	X position + (511 - horizontal count)

The X FIFO is a read-only register located at offset 0xD0. It should be accessed via 16-bit word reads, although only the lower 9 bits are significant. Each value obtained from the X FIFO is offset from the true X field-of-view position by the quantity (511 - horizontal count), where "horizontal count" is the total number of pixels per horizontal scan line. Consult the camera specifications for the correct horizontal count value.

Y FIFO (R) (0xD2)

Bits 15-9	Bits 8-0
Unused	Y position + (511 - vertical count)

The Y FIFO is a read-only register located at offset 0xD2. It should be accessed via 16-bit word reads, although only the lower 9 bits are significant. Each value obtained from the Y FIFO is offset from the

true Y field-of-view position by the quantity (511 - vertical count), where "vertical count" is the total number of horizontal scan lines. Consult the camera specifications for the correct vertical count value.

Performing the above offset calculation places the coordinate (1, 1) at the top left corner of the field of view, as viewed on a television monitor. The coordinate (horizontal count - 1, vertical count - 1) is at the lower right corner. This is a left-handed coordinate system, and vision processing software will need to take this into account.

Z0 FIFO (R) (0xD4)

Bits 15-9	Bit 8	Bits 7-0
Unused	Frame marker	Camera 0 pixel value

The Z0 FIFO is a read-only register located at offset 0xD4. It should be accessed via 16-bit word reads, although only the lower 9 bits are significant. The Z0 FIFO stores the digitized values of each bright pixel detected by the first camera (Camera 0) in the lower 8 bits. The ninth bit (Bit 8) is a frame marker. Point Grabber II sets Bit 8 of the Z0 FIFO at the end of each video frame. Because of the way most CCD cameras operate, the lower 8 bits of the Z0 FIFO will be zero when the ninth bit is 1.

Z1 FIFO (R) (0xD6)

Bits 15-9	Bit 8	Bits 7-0
Unused	Frame marker	Camera 1 pixel value

The Z1 FIFO is a read-only register located at offset 0xD6. It should be accessed via 16-bit word reads, although only the lower 9 bits are significant. The Z1 FIFO stores the digitized values of each bright pixel detected by the second camera (Camera 1) in the lower 8 bits. The ninth bit (Bit 8) is a frame marker. Point Grabber II sets Bit 8 of the Z1 FIFO at the end of each video frame. Because of the way most CCD cameras operate, the lower 8 bits of the Z0 FIFO will be zero when the ninth bit is 1. The Z1 FIFO frame marker serves the same purpose as the Z0 FIFO frame marker. Both are provided to allow software synchronization checks.

Status Register

Status Register (R) (0xD9)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Z1 Full	Z0 Full	Y Full	X Full	Z1 Empty	Z0 Empty	Y Empty	X Empty

The Status Register is a read-only byte register located at offset 0xD9. Each bit in the register indicates the full or empty status of each of the four FIFOs. Software should consult this register before performing any FIFO reads; any nonzero bit indicates an error. If it detects an error, the software should reset the FIFOs via the FIFO Reset Register (see below) to resynchronize the FIFOs.

Threshold Registers

Threshold 0 Register (W) (0xD1)

Bits 7-0
Camera 0 threshold

The Threshold 0 Register is a write-only byte register located at offset 0xD1. The 8-bit value represents a threshold voltage, from 0 V to 2 V, for Camera 0. The threshold voltage indicates the transition of the video signal from dark to bright. When the Camera 0 video signal voltage is higher than the threshold, Point Grabber II stores the X and Y scan location as well as the digitized signal strength. The threshold voltage is also the lower reference voltage of the corresponding A/D converter.

Threshold 1 Register (W) (0xD5)

Bits 7-0
Camera 1 threshold

The Threshold 1 Register is a write-only byte register located at offset 0xD5. The 8-bit value represents a threshold voltage, from 0 V to 2 V, for Camera 1. The threshold voltage indicates the transition of the video signal from dark to bright. When the Camera 1 video signal voltage is higher than the threshold, Point Grabber II stores the X and Y scan location as well as the digitized signal strength. The threshold voltage is also the lower reference voltage of the corresponding A/D converter.

ADC Limit Registers

ADC 0 Limit Register (W) (0xD3)

Bits 7-0
Camera 0 ADC limit

The ADC 0 Limit Register is a write-only byte register located at offset 0xD3. The 8-bit value represents the upper reference voltage, from 0 V to 2 V, for the Camera 0 A/D converter. The setting of the upper reference voltage alters the sensitivity of Point Grabber II. By setting ADC 0 Limit to correspond to the highest expected voltage of the Camera 0 video signal, one can maximize sensitivity.

ADC 1 Limit Register (W) (0xD7)

Bits 7-0
Camera 1 ADC limit

The ADC 1 Limit Register is a write-only byte register located at offset 0xD7. The 8-bit value represents the upper reference voltage, from 0 V to 2 V, for the Camera 1 A/D converter. The setting of the upper reference voltage alters the sensitivity of Point Grabber II. By setting ADC 1 Limit to correspond to the highest expected voltage of the Camera 1 video signal, one can maximize sensitivity.

Enable Register

Enable Register (R/W) (0xDB)

Bit 7	Bits 6-2	Bit 1	Bit 0
Interrupt Enable	Unused	Camera 1 Enable	Camera 0 Enable

The Enable Register is a read-write byte register located at offset 0xDB. Logical 1 indicates enable. One can individually control data collection from each camera via the corresponding enable bits. When a camera is "disabled", only the data storage for that camera is disabled. The camera is still operational, and the video signal can still be viewed on a television monitor. Interrupt generation also may be enabled or disabled using bit 7 of the Enable Register. Bits 6 through 2 are unused.

FIFO Reset Register

FIFO Reset Register (W) (0xDD)

Bits 7-0
Don't Care

The FIFO Reset Register is a write-only byte register located at offset 0xDD. Any writes to this address location will reset and empty all FIFOs. The value written is irrelevant.

The FIFOs need to be reset immediately after enabling interrupts to ensure that the data will be synchronized. They should also be reset after an error has been detected.

Board Reset Register

Board Reset Register (W) (0xDF)

Bits 7-0
Don't Care

The Board Reset Register is a write-only byte register located at offset 0xDF. Any writes to this address location will reset Point Grabber II. The value written is irrelevant. Resetting the board brings Point Grabber II to the power-on condition, which means that the cameras are disabled, the FIFOs are reset, the threshold and ADC limit voltages are zero, interrupt is disabled, and the interrupt level and vector are zero.

Interrupt Control Register

Interrupt Control Register (R/W) (0xE1)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F	FAC	X/IN*	IRE	IRAC	L2	L1	L0

The Interrupt Control Register is a read-write byte register located at offset 0xE1. This register controls the interrupt level, a separate interrupt enable, and fields that determine actions during an interrupt acknowledge cycle. The fields are defined as follows:

1. Interrupt level (L2, L1, L0) — The least significant 3-bit field of the register determines the level at which an interrupt will be generated:

L2	L1	L0	IRQ Level
0	0	0	Disabled
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

A value of zero in the field disables the interrupt.

2. Interrupt Enable (IRE) — This field (Bit 4) must be set (high level) to enable the bus interrupt request.

3. Interrupt Auto-Clear (IRAC) — If the IRAC is set (Bit 3), IRE (Bit 4) is cleared during an interrupt acknowledge cycle responding to this request. This action of clearing IRE disables further interrupt request. To re-enable the interrupt associated with this register, IRE must be set again by writing to the control register. Leave this bit unset during normal operations.

4. External/Internal (X/IN*) — Always clear (low level) this bit for Point Grabber II.

5. Flag (F) — Bit 7 is a flag that can be used in conjunction with the test and set instruction of the MC680xx family of microprocessors. It can be changed without affecting Point Grabber II operation. This flag may be useful for processor-to-processor communication and resource allocation (i.e., symaphors). This flag is typically not used.

6. Flag Auto-Clear (FAC) — If FAC (Bit 6) is set, the Flag bit is automatically cleared during an interrupt acknowledge cycle.

Interrupt Vector Register

Interrupt Vector Register (R/W) (0xE3)

Bits 7-0
Interrupt Vector

The Interrupt Vector Register is a read-write byte register located at offset 0xE3. This 8-bit interrupt vector is supplied, during an interrupt acknowledge cycle, to the CPU for calculating the location of the interrupt service routine. For the VMEbus, the routine is located at an address that is four (4) times the vector value.

B.2 Resolution Tests

Figures B.2 and B.3 show the steady-state noise characteristics of the Point Grabber II vision system. The data was taken from the global vision system viewing a single LED. The vision system covers a field of view of 2.4m in the x direction, and 1.7m in y direction. The noise is about .2mm in the x direction and about .1mm in the y direction—a resolution of better than 1 in 10,000.

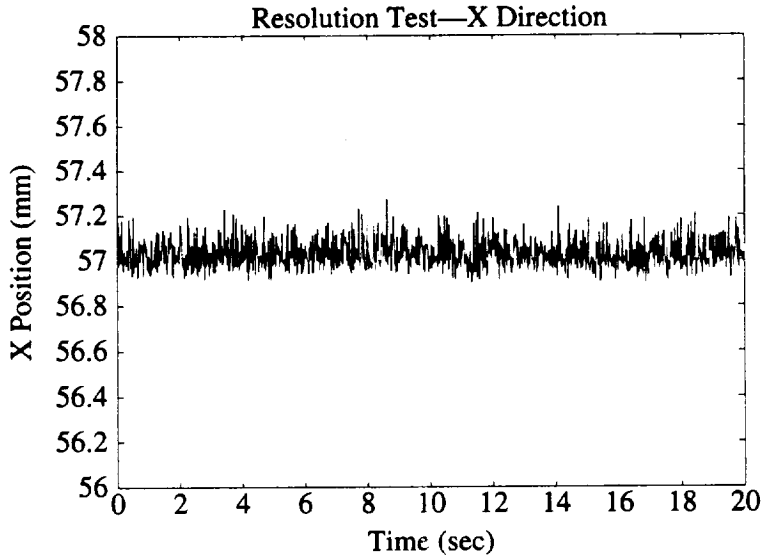


Figure B.2: Noise Characteristics—X Direction

Figures B.4 and B.5 show the resolution of the vision system by demonstrating the transient behavior. The LED, mounted on one of the free-flying payload objects, is moved slowly and smoothly along a diagonal. The plots show relatively smooth, monotonic functions of time for the LED x and y positions. Although there are no sharp jumps at camera-pixel boundaries, the pixel effect can still be seen as deviations from a straight line in the plots. The "bumps" in the x direction occur at about every 5mm, and those in the y direction occur at about every 8mm.

There are 380 active camera pixels in the x direction and 190 active pixels in the y direction. The size of each pixel, therefore, is calculated to be:

$$x_{size} = 2400\text{mm}/380 = 6.6\text{mm}$$

$$y_{size} = 1700\text{mm}/190 = 8.9\text{mm}$$

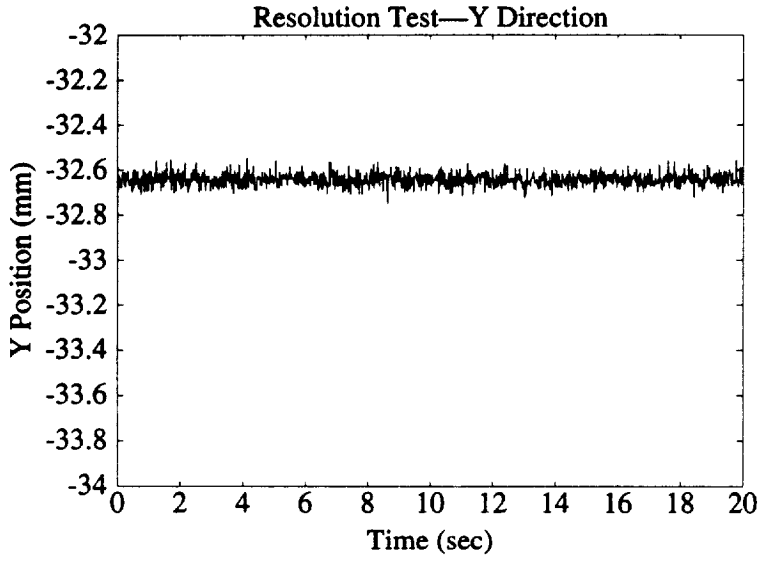


Figure B.3: Noise Characteristics—Y Direction

This agrees reasonably well with the observed behavior in Figures B.4 and B.5. Larger targets or LED's that cover more camera pixels will alleviate this effect.

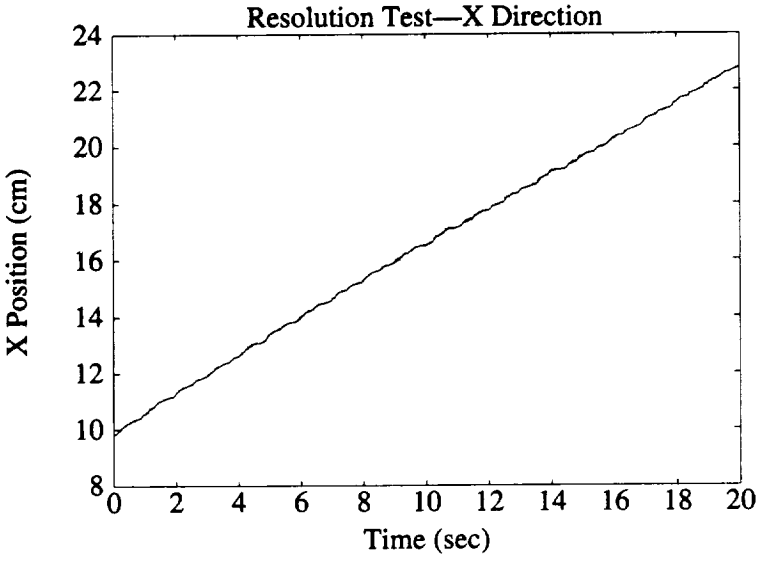


Figure B.4: **Resolution Test—X Direction**

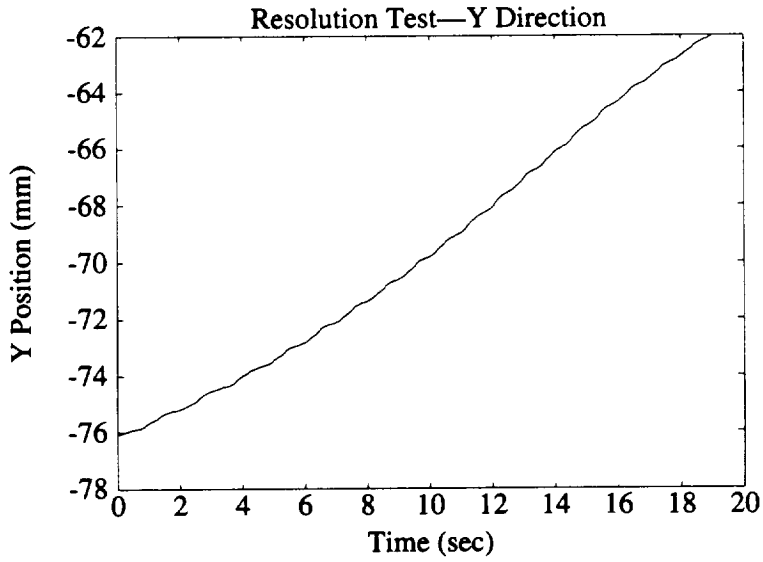
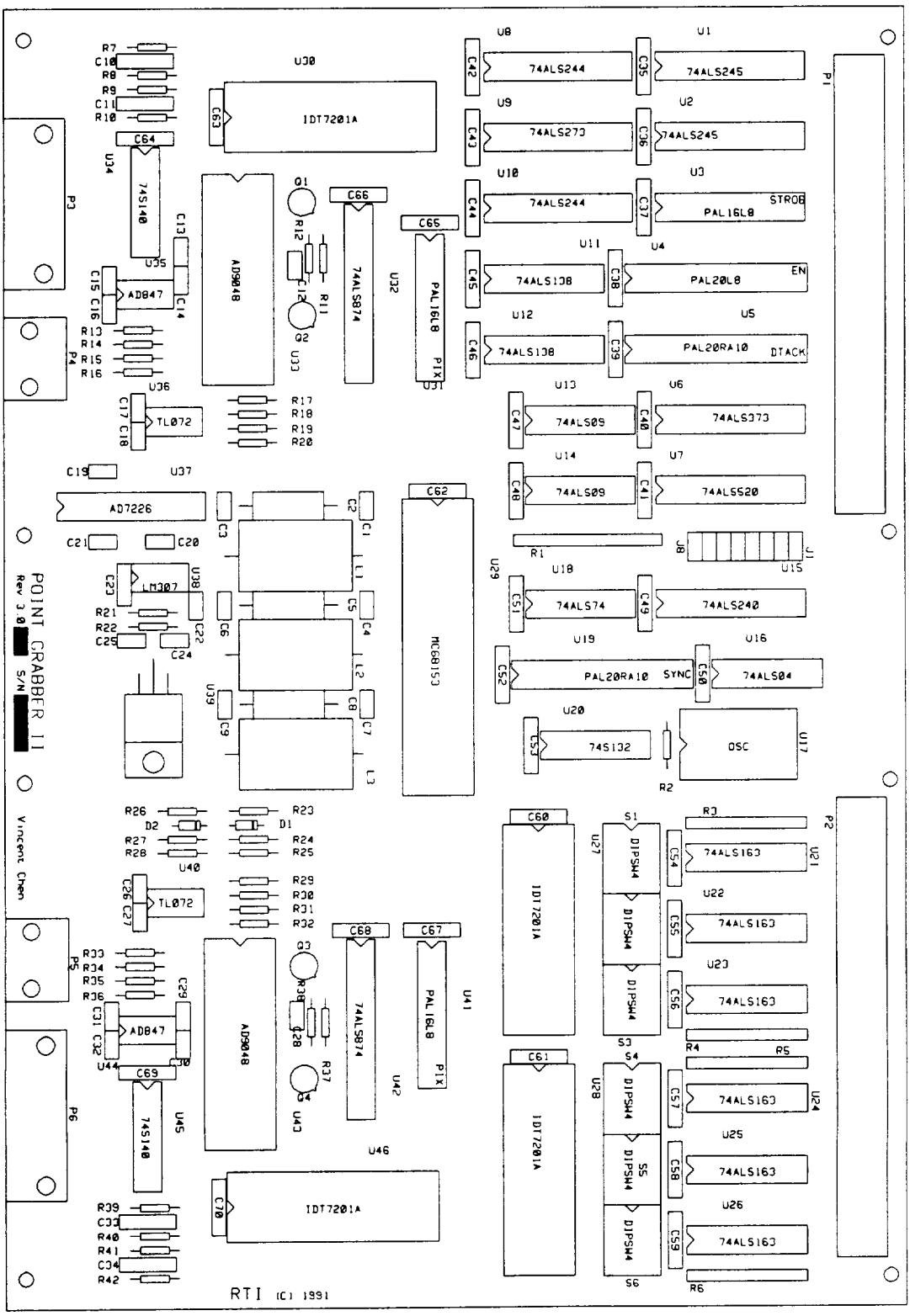
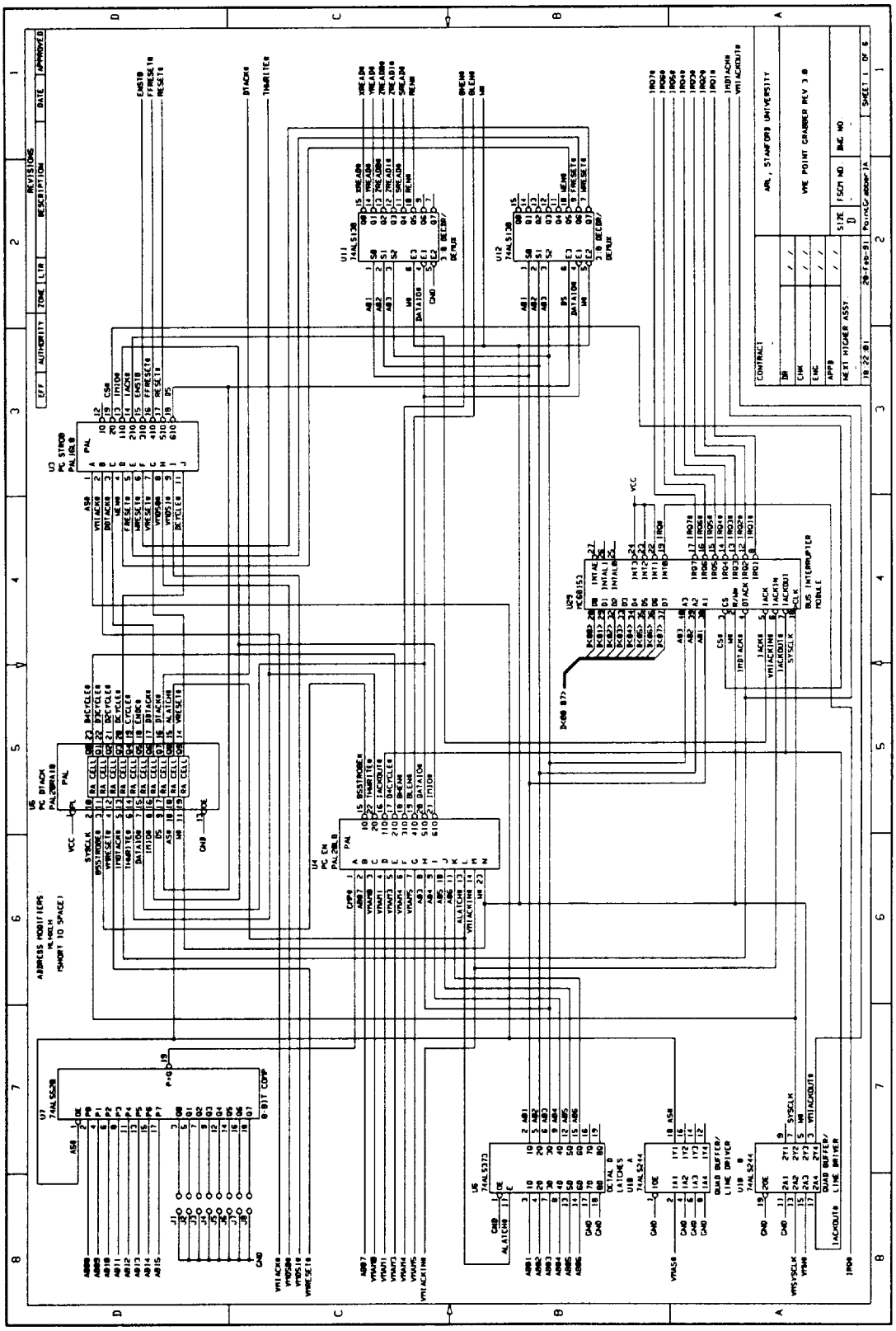


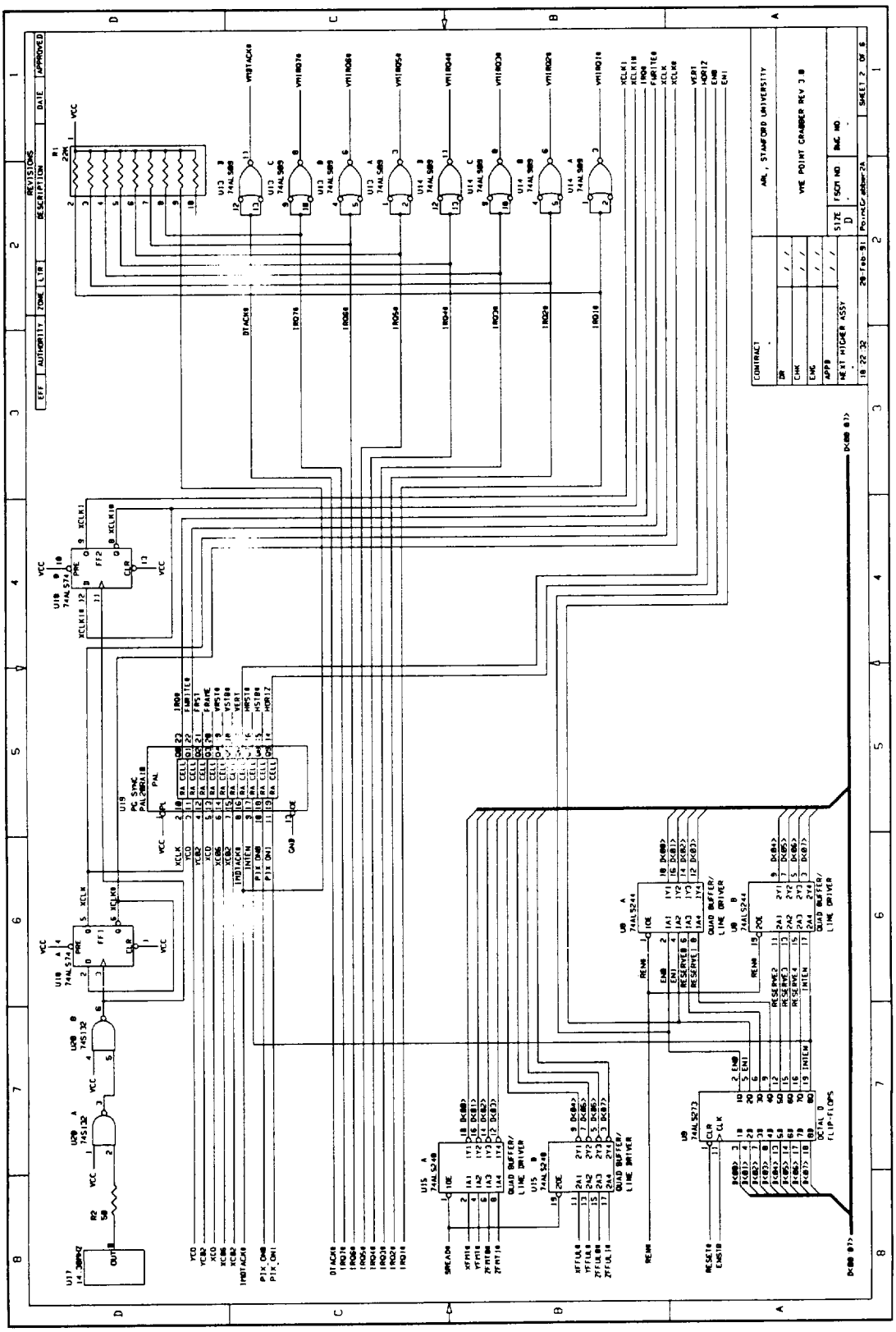
Figure B.5: **Resolution Test—Y Direction**

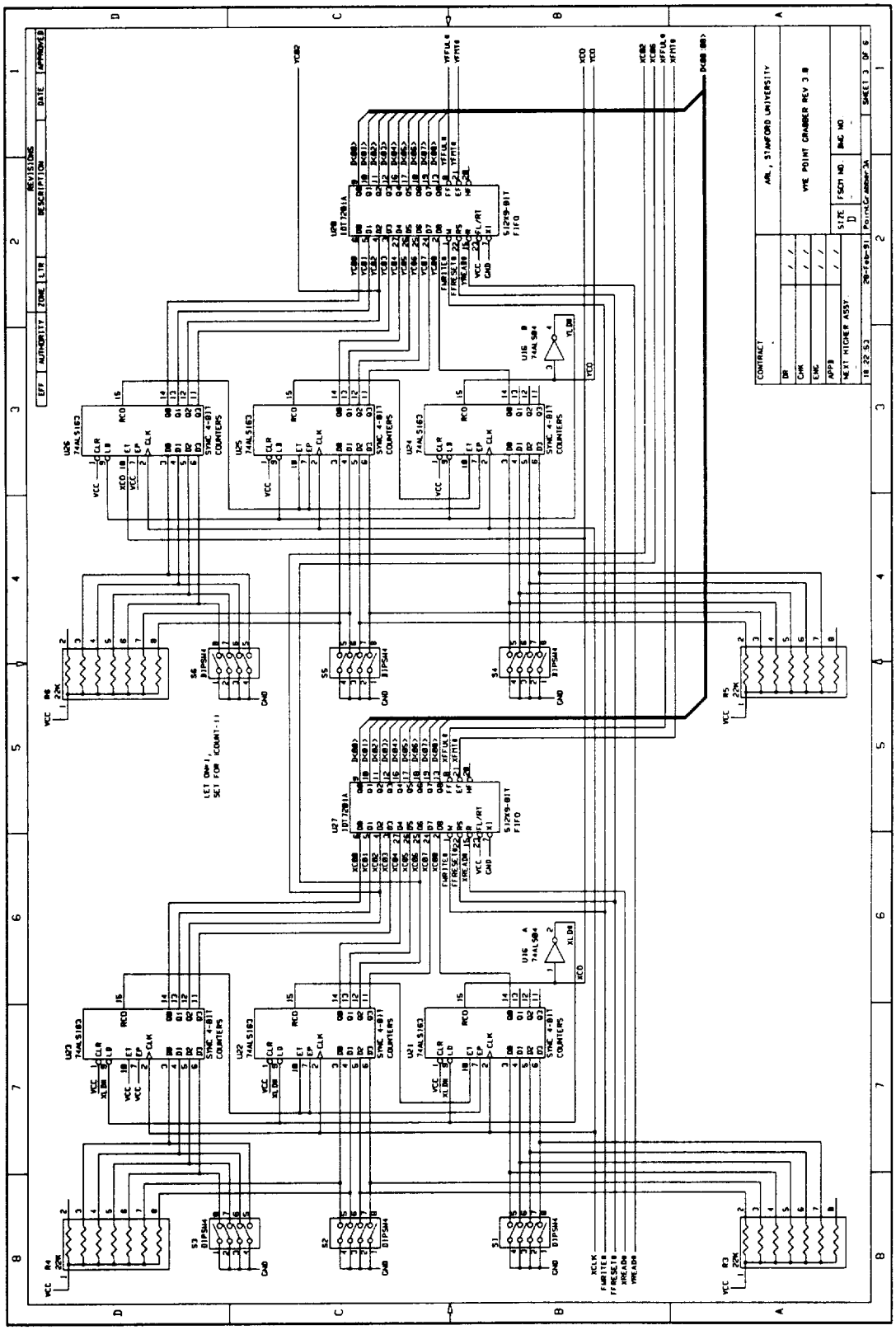
B.3 Schematics

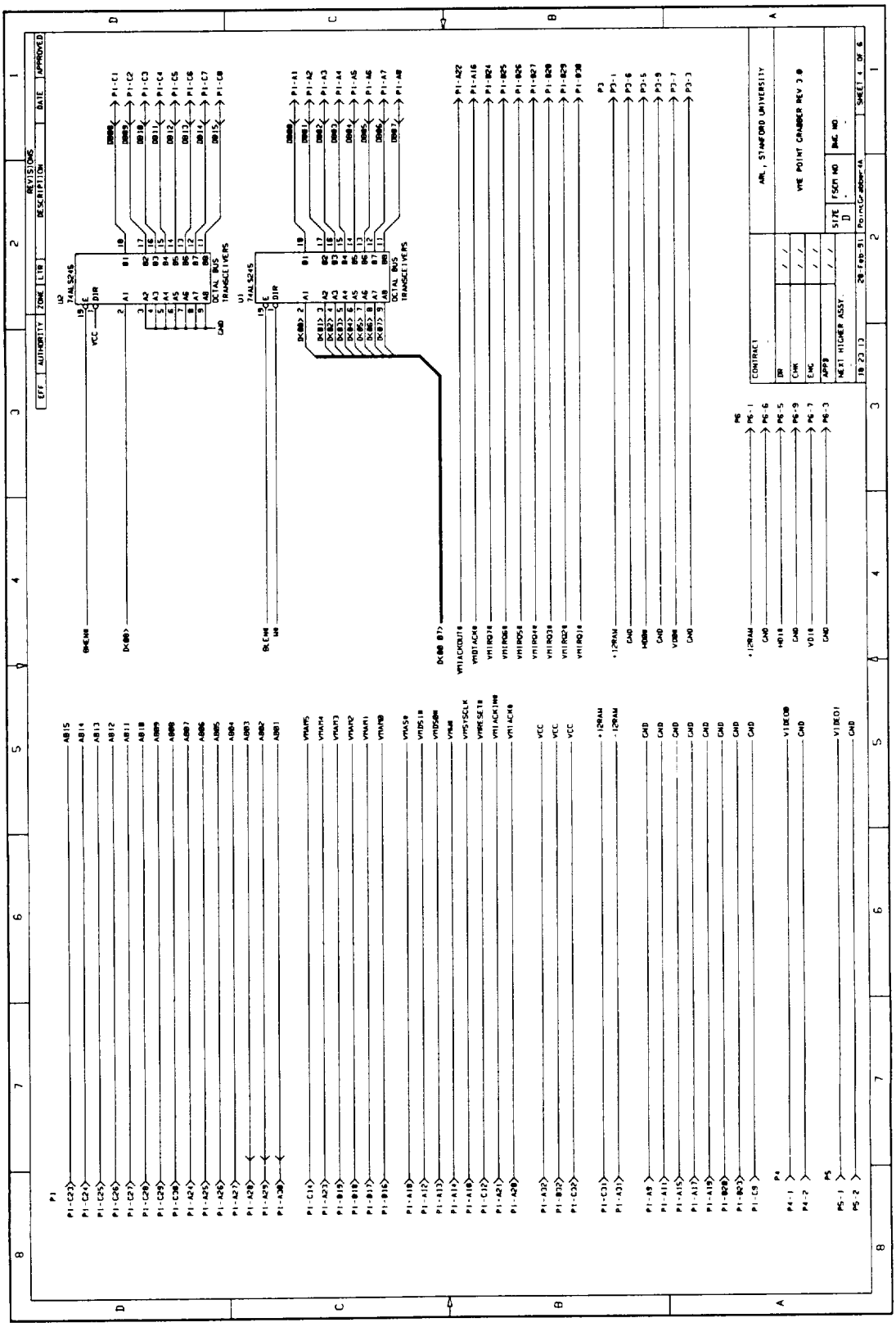
The schematics start on the following page.

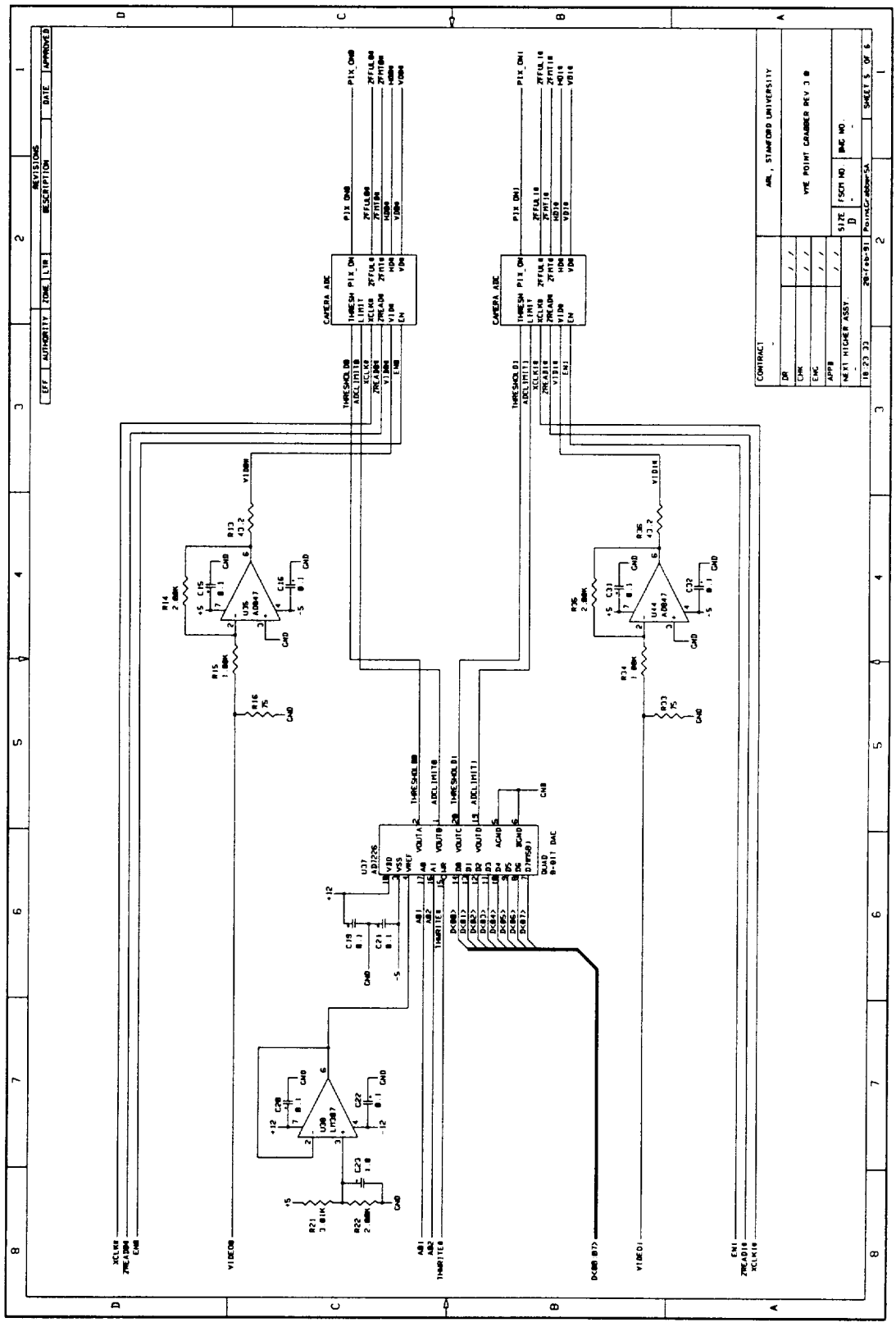

POINT GRABBER II
Rev 3.0
S/N
Vincent Chen
RTI (C) 1991
IDT7201A
74ALS244
74ALS245
74ALS273
74ALS138
PAL16L8
PAL20L8
PAL20RA10
74ALS09
74ALS373
74ALS20
74ALS74
74ALS240
74ALS04
74S132
OSC
74ALS163
AD9048
74ALS874
AD847
TL072
74S140
AD7226
LM307
MC68153
DIPSW4

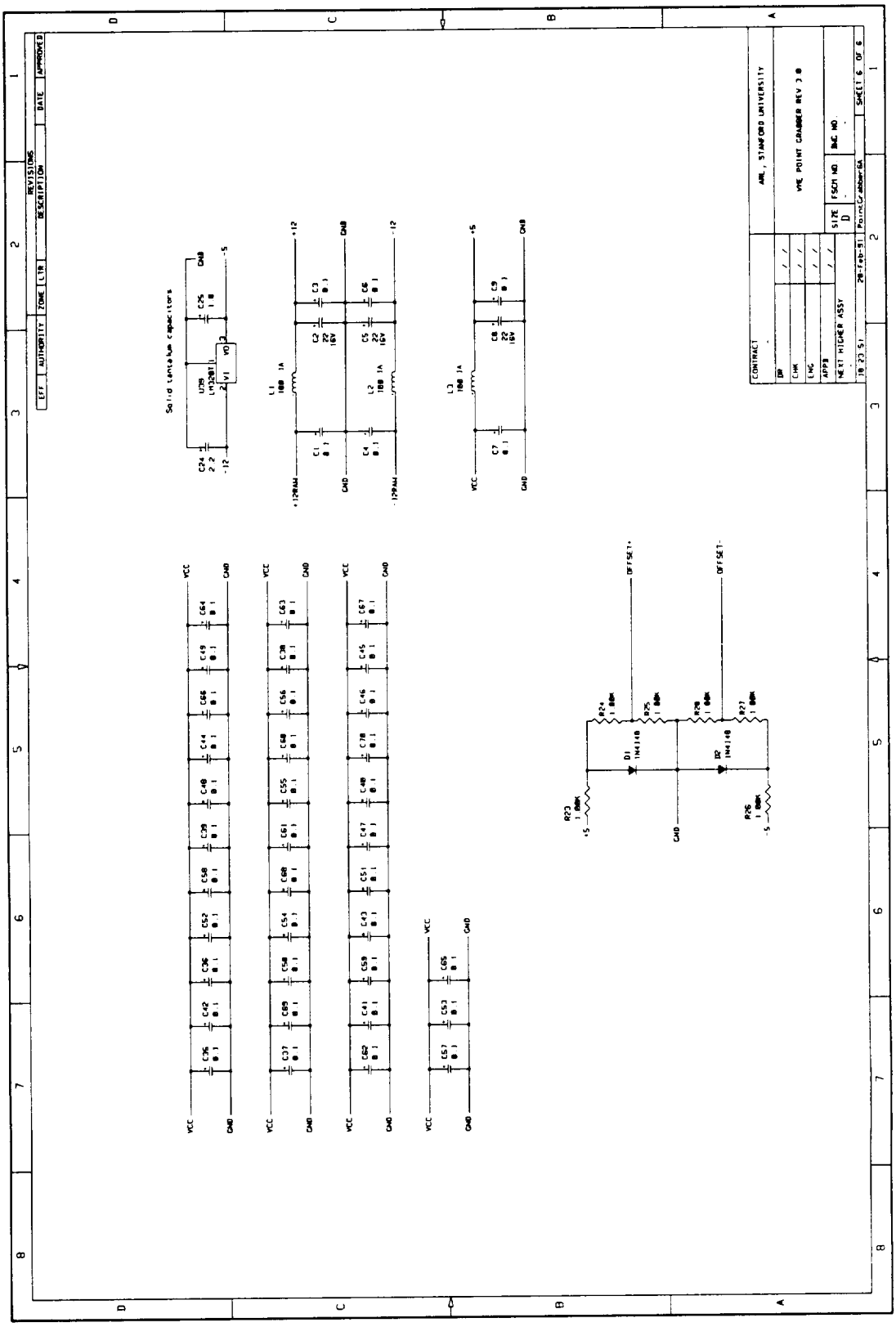


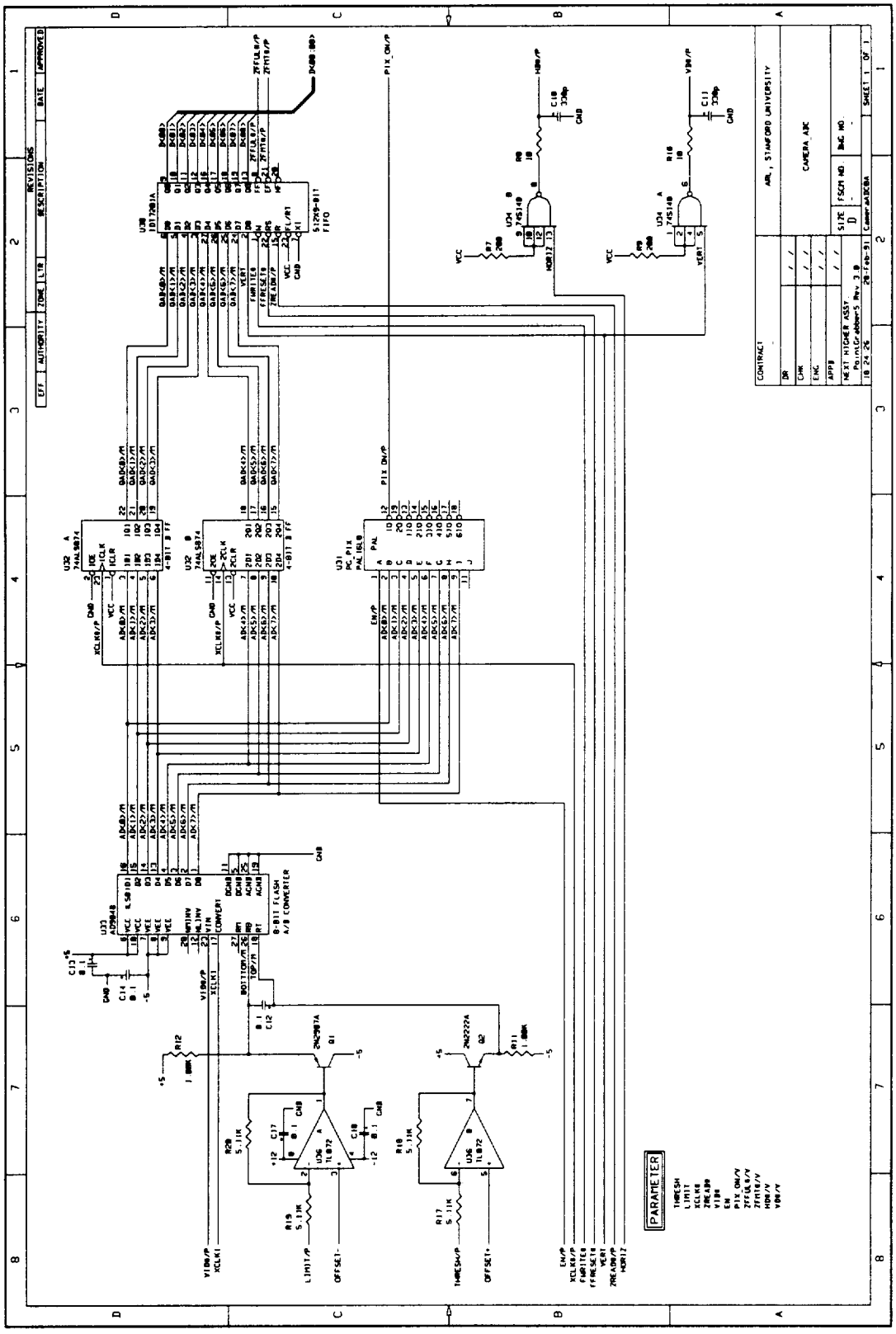












B.4 PALASM Listings

B.4.1 DTACK Timing

```
TITLE       PG DTACK TIMING
PATTERN     1
REVISION    3.0B
AUTHOR      VINCENT CHEN
COMPANY     ARL, STANFORD UNIVERSITY
DATE        09-30-90

; This Pal generates several handshaking signals required for the interface
; between the Point Grabber and the VME Bus:
;     /VRESET is the VME reset signal, synchronized to SYSCLK.
;     /ALATCH is the address latch signal.  BSSTROBE initiates the latch.
; The latch is terminated by DS and AS going invalid during a READ cycle
; (ENDC) indicating that the VME Bus has completed the access, so we no longer
; need to address our board registers.  On a WRITE cycle, the latch is
; terminated by our DTACK going valid, indicating that we will be ready
; for the next access (almost) immediately after acknowledging the WRITE.
;     /DTACK is the data acknowledge signal the board sends the VME Bus
; indicating that the data is ready on READ cycles or that the data has
; been stored on WRITE cycles.  The basic timing for /DTACK is through
; the CYCLE and DxCYCLE signals.  Together, these guarantee a minimum of
; one (1) SYSCLK cycle (62.5 ns) between addressing board registers
; (DATAIO or IMIO valid) and lowering /DTACK, or four (4) SYSCLK cycles
; (250 ns) between writing the threshold D/A and lowering /DTACK.  This
; stupid D/A (AD7226) requires a write pulse of a minimum of 200 ns.  DDTACK
; is the data acknowledge from accessing the the data registers,
; while IMDTACK is the data acknowledge generated by the Interrupt
; Module chip.  The CYCLE, DCYCLE, etc. registers are cleared by ALATCH
; going invalid.

; Bug fixes: added ALATCH to ENDC
;               modified ENDC to deassert ALATCH immediate after DTACK on
;               WRITES


CHIP        PALPG_DTACK          PAL20RA10

PIN     1   PL          LOW             ;I
PIN     2   SYSCLK      HIGH            ;I
PIN     3   BSSTROBE    LOW             ;I
PIN     4   VMRESET     LOW             ;I
PIN     5   IMDTACK     LOW             ;I
PIN     6   THWRITE     LOW             ;I
PIN     7   DATAIO      LOW             ;I
PIN     8   IMIO        LOW             ;I
PIN     9   DS          HIGH            ;I
PIN     10  AS          LOW             ;I
PIN     11  W           LOW             ;I
PIN     12  GND

PIN     13  OE          LOW             ;I
```

```
PIN     14  VRESET      REG     LOW     ;O
PIN     15  ALATCH      REG     LOW     ;O
PIN     16  DTACK       LOW             ;O
PIN     17  DDTACK      LOW             ;O
PIN     18  ENDC        LOW             ;O
PIN     19  CYCLE       REG     LOW     ;O
PIN     20  DCYCLE      REG     LOW     ;O
PIN     21  D2CYCLE     REG     LOW     ;O
PIN     22  D3CYCLE     REG     LOW     ;O
PIN     23  D4CYCLE     REG     LOW     ;O
PIN     24  VCC

EQUATIONS

; VRESET IS A SYNCHRONIZED VERSION OF VME RESET (VMRESET)

VRESET = VMRESET
VRESET.CLKF = SYSCLK

; ADDRESS LATCH IS TRIGGERED BY BSSTROBE AND CLEARED BY ENDC

ALATCH = VCC
ALATCH.CLKF = BSSTROBE
ALATCH.RSTF = ENDC

; DATA ACKNOWLEDGE FROM THE DATA SECTION. IF NOT TO THRESHOLD WRITES, THEN
; ONLY ONE SYSCLK DELAY.  IF THRESHOLD WRITE, THEN 4 SYSCLK DELAYS.

DDTACK = DCYCLE * DATAIO * /THWRITE
       + D4CYCLE * THWRITE

; DATA ACKNOWLEDGE TO THE VME BUS IS COMPRISED OF DTACKS FROM BOTH THE DATA
; SECTION AND THE INTERRUPT MODULE CHIP

DTACK = DCYCLE * DATAIO * /THWRITE
      + D4CYCLE * THWRITE
      + IMDTACK

; END CONDITION FOR A BOARD ACCESS CYCLE IS DS AND AS INVALID ON READ CYCLES,
; DTACK VALID ON WRITE CYCLES, OR VME RESET

ENDC = /DS * /AS * ALATCH * /W
    + DTACK * W
    + VRESET

; CYCLE TIMING IS INITIATED WHEN EITHER THE DATA SECTION OR INTERRUPT MODULE
; IS BEING ACCESSED.  BY SYNCHRONIZING TO SYSCLK, DCYCLE IS GUARANTEED TO BE
; ASSERTED A MINIMUM OF ONE SYSCLK CYCLE AND A MAXIMUM OF TWO SYSCLK CYCLES
; AFTER THE BOARD REGISTERS HAVE BEEN ADDRESSED.  CYCLE AND DCYCLE ARE
; CLEARED WHEN ALATCH IS INVALID.  THE 4 TIMES DELAYED CYCLE SIGNAL, D4CYCLE,
; IS USED BY THE DAC ON WRITES, WHICH NEEDS 200NS.  D4CYCLE IS GUARANTEEING
; A MINIMUM OF 250NS.

CYCLE = DS * DATAIO + DS * IMIO        ; NOTE THAT DS MUST ALSO BE VALID
CYCLE.CLKF = SYSCLK
```

```
CYCLE.RSTF = /ALATCH

DCYCLE = CYCLE
DCYCLE.CLKF = SYSCLK
DCYCLE.RSTF = /ALATCH

D2CYCLE = DCYCLE
D2CYCLE.CLKF = SYSCLK
D2CYCLE.RSTF = /ALATCH

D3CYCLE = D2CYCLE
D3CYCLE.CLKF = SYSCLK
D3CYCLE.RSTF = /ALATCH

D4CYCLE = D3CYCLE
D4CYCLE.CLKF = SYSCLK
D4CYCLE.RSTF = /ALATCH

SIMULATION

TRACE_ON SYSCLK DS DATAIO IMIO THWRITE IMDTACK
VMRESET VRESET BSSTROBE ALATCH ENDC CYCLE DCYCLE D4CYCLE DDTACK DTACK

SETF /PL OE /SYSCLK /BSSTROBE /DS /DATAIO /IMIO /THWRITE /IMDTACK /VMRESET

PRLDF /ALATCH /CYCLE /DCYCLE /D2CYCLE /D3CYCLE /D4CYCLE /VRESET

; FIRST TRY DATA SECTION ACCESS

SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK

SETF BSSTROBE                                   ; RIGHT ACCESS
CHECK ALATCH                                    ; CHECK FOR ALATCH VALID

SETF DS                                         ; DATA STROBE
SETF DATAIO                                     ; DATA ACCESS

FOR I := 1 TO 4 DO                              ; WAIT 4 CLOCKS
  BEGIN
    SETF SYSCLK
    SETF /SYSCLK
;   CLOCKF SYSCLK
  END

SETF /DS
SETF /DATAIO
SETF /BSSTROBE
SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK

; NOW TRY THRESHOLD WRITES
```

```
SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK

SETF BSSTROBE                          ; RIGHT ACCESS
CHECK ALATCH                           ; CHECK FOR ALATCH VALID

SETF DATAIO THWRITE                    ; DATA ACCESS
SETF DS                                ; DATA STROBE

FOR I := 1 TO 6 DO                     ; WAIT 4 CLOCKS
  BEGIN
    SETF SYSCLK
    SETF /SYSCLK
;    CLOCKF SYSCLK
  END

SETF /DS
SETF /DATAIO /THWRITE
SETF /BSSTROBE
SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK

; NOW TRY IM ACCESS
SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK                         ; MAKE CLOCK HIGH

SETF BSSTROBE                          ; RIGHT ACCESS
CHECK ALATCH                           ; CHECK FOR ALATCH VALID

SETF DS                                ; DATA STROBE
SETF IMIO                              ; IM ACCESS

FOR I := 1 TO 4 DO                     ; WAIT 4 CLOCKS
  BEGIN
    SETF SYSCLK
    SETF /SYSCLK
;    CLOCKF SYSCLK
  END

SETF IMDTACK                           ; IM'S DTACK
SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK

SETF /BSSTROBE
SETF /DS
SETF /IMIO
SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK
SETF /IMDTACK
```

```
; CHECK VRESET

SETF VMRESET                                    ; RAISE RESET LINE
SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK
SETF SYSCLK
SETF /SYSCLK
;CLOCKF SYSCLK

TRACE_OFF
```

B.4.2 Buffer Enables and Device Select

```
TITLE       PG BUFFER ENABLES AND DEVICE SELECT
PATTERN     1
REVISION    2.0+
AUTHOR      VINCENT CHEN
COMPANY     ARL, STANFORD UNIVERSITY
DATE        2-08-90

; This Point Grabber PAL generates enables for the data buffers (/BHEN,
; /BLEN), device selects (/DATAIO, /IMIO, /THWRITE), and the board select
; strobe (/BSSTROBE).
;     /DATAIO indicates access to the data section of the Point Grabber.
;     /IMIO indicates access to the Interrupt Module.
;     /BHEN is the enable for the high byte of data, and it is asserted
; only when access is to the data section on a read cycle.
;     /BLEN is the enable for the low byte of data, and it is asserted
; when access is to either the data or IM sections and also during an
; interrupt cycle.  When the processor is responding to an interrupt from
; the Point Grabber, the Point Grabber will receive the daisy-chained
; /VMIACKIN signal, but it will not assert /IACKOUT, thus enabling the
; data buffer.  The Interrupt module will then send out the interrupt
; vector onto the VME Bus.
;     BSSTROBE is the board select strobe.  /CMP is the output of the 8-bit
; comparator comparing A08-A15, and VMAMx are the address modifier bits.  The
; board is presently configured to fit in short IO space, occupying a 128 byte
; block.  The location of the block is determined by the input to the
; comparator generating /CMP.

CHIP        PALPG_EN            PAL20L8

PIN     1   CMP         LOW     ;I
PIN     2   AB07        HIGH    ;I
PIN     3   VMAM0       HIGH    ;I
PIN     4   VMAM1       HIGH    ;I
PIN     5   VMAM3       HIGH    ;I
PIN     6   VMAM4       HIGH    ;I
PIN     7   VMAM5       HIGH    ;I
PIN     8   A03         HIGH    ;I
PIN     9   A04         HIGH    ;I
PIN    10   A05         HIGH    ;I
```

```
PIN     11    A06          HIGH     ;I
PIN     12    GND

PIN     13    ALATCH       LOW      ;I
PIN     14    VMIACKIN     LOW      ;I
PIN     15    BSSTROBE     LOW      ;O
PIN     16    IACKOUT      LOW      ;I
PIN     17    D4CYCLE      LOW      ;I
PIN     18    BHEN         LOW      ;O
PIN     19    BLEN         LOW      ;O
PIN     20    DATAIO       LOW      ;O
PIN     21    IMIO         LOW      ;O
PIN     22    THWRITE      LOW      ;O
PIN     23    W            LOW      ;I
PIN     24    VCC


EQUATIONS

; BOARD SELECT HAS HIGH 128 BYTES OF A 256 BYTE BLOCK IN SHORT IO SPACE

BSSTROBE = CMP * AB07 * VMAM0 * /VMAM1 * VMAM3 * /VMAM4 * VMAM5
                * A06 * /A05 * A04
         + CMP * AB07 * VMAM0 * /VMAM1 * VMAM3 * /VMAM4 * VMAM5
                * A06 * A05 * /A04

BHEN = /W * ALATCH * A06 * /A05 * A04

BLEN = /IACKOUT * VMIACKIN
     + ALATCH * A06 * /A05 * A04
     + ALATCH * A06 * A05 * /A04

DATAIO = ALATCH * A06 * /A05 * A04

IMIO = ALATCH * A06 * A05 * /A04

THWRITE = W * ALATCH * A06 * /A05 * A04 * /A03 * /D4CYCLE

SIMULATION

TRACE_ON CMP AB07 VMAM0 VMAM1 VMAM3 VMAM4 VMAM5 BSSTROBE
A06 A05 A04 A03 ALATCH W VMIACKIN IACKOUT
BLEN BHEN DATAIO IMIO D4CYCLE THWRITE

SETF /CMP AB07 VMAM0 VMAM1 VMAM3 VMAM4 VMAM5
/ALATCH A06 A05 A04 A03 W /VMIACKIN /IACKOUT /D4CYCLE

; TEST BSSTROBE

SETF CMP
SETF /VMAM1
SETF /VMAM4
CHECK /BSSTROBE
SETF /A05
CHECK BSSTROBE
```

```
SETF A05 /A04
CHECK BSSTROBE
SETF A04
SETF /CMP VMAM1 VMAM4

; TEST BLEN

SETF VMIACKIN
CHECK BLEN
SETF IACKOUT
CHECK /BLEN
SETF /VMIACKIN /IACKOUT

SETF A06 /A05 A04                    ; ALSO TEST DATAIO
SETF ALATCH
CHECK BLEN DATAIO
SETF /ALATCH A05

SETF A06 A05 /A04                    ; ALSO TEST IMIO
SETF ALATCH
CHECK BLEN IMIO
SETF /ALATCH A04

; TEST BHEN

SETF /W
SETF A06 /A05 A04                    ; ALSO TEST DATAIO
SETF ALATCH
CHECK BHEN
SETF /ALATCH W A05

; TEST THWRITE

SETF /W
SETF A06 /A05 A04 /A03
SETF W
SETF ALATCH
CHECK THWRITE
SETF /CMP AB07 VMAM1 VMAM4
SETF D4CYCLE
CHECK /THWRITE
SETF /ALATCH
SETF A05 A03 /D4CYCLE
SETF /W

TRACE_OFF
```

B.4.3 Pixel Detect

```
TITLE       PG PIXEL DETECTOR
PATTERN     1
REVISION    3.0+
AUTHOR      Vincent Chen
```

```
COMPANY     Stanford University, ARL
DATE        07-21-90

; This Point Grabber PAL generates the PIXel_ON signal to trigger FIFO
; writes.  It takes as input the ADC output and the Enable signal
; for a camera and generates the PIX_ON signal

CHIP            PALPG_PIX       PAL16L8

PIN     1       EN              HIGH    ;I
PIN     2       AD0             HIGH    ;I
PIN     3       AD1             HIGH    ;I
PIN     4       AD2             HIGH    ;I
PIN     5       AD3             HIGH    ;I
PIN     6       AD4             HIGH    ;I
PIN     7       AD5             HIGH    ;I
PIN     8       AD6             HIGH    ;I
PIN     9       AD7             HIGH    ;I
PIN    10       GND

PIN    11       NC
PIN    12       PIX_ON          HIGH    ;O
PIN    13       NC
PIN    14       NC
PIN    15       NC
PIN    16       NC
PIN    17       NC
PIN    18       NC
PIN    19       NC
PIN    20       VCC

EQUATIONS

/PIX_ON = /AD0 * /AD1 * /AD2 * /AD3 * /AD4 * /AD5 * /AD6 * /AD7
        + /EN                           ;Need the enable signal

SIMULATION

TRACE_ON EN AD0 AD1 AD2 AD3 AD4 AD5 AD6 AD7 PIX_ON

SETF /EN /AD0 /AD1 /AD2 /AD3 /AD4 /AD5 /AD6 /AD7    ;init

;test PIX_ON signal
SETF AD0
CHECK /PIX_ON
SETF /AD0
SETF EN
SETF AD0
CHECK PIX_ON
SETF /AD0
CHECK /PIX_ON
SETF AD7
CHECK PIX_ON
SETF /AD7
SETF /EN
```

```
TRACE_OFF
```

B.4.4 Data Strobe and Resets

```
TITLE       PG STROBES AND RESETS
PATTERN     1
REVISION    2.0+
AUTHOR      VINCENT CHEN
COMPANY     ARL, STANFORD UNIVERSITY
DATE        2-08-90

; This Point Grabber PAL generates the strobe signals for the camera enables
; and reset signals, as well as some auxiliary signals required to
; interface with the VME Bus.
;
;     IACK is the interrupt acknowledge from the bus master.  It is also
; asserted, along with CS to reset the interrupt module.
;     ENSTB is the strobe for the camera/interrupt enable latch.
;     /FFRESET is the reset for the FIFOs.
;     /RESET is the reset for the threshold latches.
;     DS is the data strobe for the Point Grabber, comprising of VME's
; DS1 and DS0.
;     /CS is the chip select for the Interrupt Module.  It is asserted
; when DCYCLE becomes valid, allowing time for data to stabilize.  On write
; cycles, the leading edge of /CS makes the IM latch the data, and on read
; cycles, /CS signals a data acknowledge to the IM.  The Interrupt Module can
; be reset by asserting the /IACK and /CS lines, thus VME resets (VRESET) and
; software resets (WRESET) need to assert /CS.

CHIP        PALPG_STROBE          PAL16L8

PIN     1   AS          LOW       ;I
PIN     2   VMIACK      LOW       ;I
PIN     3   DDTACK      LOW       ;I
PIN     4   WEN         LOW       ;I
PIN     5   FRESET      LOW       ;I
PIN     6   WRESET      LOW       ;I
PIN     7   VRESET      LOW       ;I
PIN     8   VMDS0       LOW       ;I
PIN     9   VMDS1       LOW       ;I
PIN    10   GND

PIN    11   DCYCLE      LOW       ;I
PIN    12   NC
PIN    13   IMIO        LOW       ;I
PIN    14   IACK        LOW       ;O
PIN    15   ENSTB       HIGH      ;O
PIN    16   FFRESET     LOW       ;O
PIN    17   RESET       LOW       ;O
PIN    18   DS          HIGH      ;O
PIN    19   CS          LOW       ;O
PIN    20   VCC
```

```
EQUATIONS

IACK = VMIACK * AS + VRESET + WRESET

/ENSTB = /WEN + /DDTACK

FFRESET = VRESET + WRESET + FRESET

RESET = VRESET + WRESET

/DS = /VMDS0 * /VMDS1

CS = DCYCLE * IMIO + VRESET + WRESET

SIMULATION

TRACE_ON DDTACK WEN ENSTB AS VMIACK WRESET VRESET FRESET FFRESET RESET IACK
VMDS0 VMDS1 DS DCYCLE IMIO CS


SETF /WEN /DDTACK /AS /VMIACK /WRESET /VRESET /FRESET /VMDS0
/VMDS1 /DCYCLE /IMIO                    ; INITIALIZE

SETF WEN
CHECK /ENSTB
SETF DDTACK                             ; WRITE ENABLE REGISTER
CHECK ENSTB
SETF /DDTACK /WEN
CHECK /ENSTB


; TEST IACK

SETF VMIACK
SETF AS
CHECK IACK
SETF /VMIACK
CHECK /IACK
SETF /AS

; NOW TEST RESETS

CHECK /FFRESET /RESET /CS
SETF VRESET                             ; VME RESET
CHECK FFRESET RESET CS
SETF /VRESET

SETF WRESET                             ; BOARD RESET
CHECK FFRESET RESET CS
SETF /WRESET

SETF FRESET                             ; FIFO RESET
CHECK FFRESET /RESET /CS
```

```
SETF /FRESET

CHECK /DS
SETF VMDS0                              ; TEST DS
CHECK DS
SETF /VMDS0
SETF VMDS1
CHECK DS
SETF /VMDS1
CHECK /DS

SETF IMIO
SETF DCYCLE                             ; NOW TEST CS
CHECK CS
SETF /DCYCLE
CHECK /CS

SETF /IMIO
TRACE_OFF
```

B.4.5 Horizontal and Vertical Syncs

```
TITLE       PG SYNC GENERATOR, INTERRUPT AND FIFO WRITE
PATTERN     1
REVISION    3.0+
AUTHOR      VINCENT CHEN
COMPANY     ARL, STANFORD UNIVERSITY
DATE        07-21-90

; Requires 20ns PAL, or SEEQ PQ20RA10Z-35

; This Point Grabber PAL provides the sync signals to the cameras as well
; as the interrupt signals at the end of each frame and the write pulse
; to the FIFOs at each edge and at the end of each frame.

CHIP        PALPG_SYNC          PAL20RA10

PIN     1   PL          LOW         ;I
PIN     2   XCLK        HIGH        ;I
PIN     3   YCO         HIGH        ;I
PIN     4   YC02        HIGH        ;I
PIN     5   XCO         HIGH        ;I
PIN     6   XC06        HIGH        ;I
PIN     7   XC02        HIGH        ;I
PIN     8   IMDTACK     LOW         ;I
PIN     9   INTEN       HIGH        ;I
PIN    10   PIX_ON0     HIGH        ;I
PIN    11   PIX_ON1     HIGH        ;I
PIN    12   GND

PIN    13   OE          LOW                 ;I
PIN    14   HORIZ       REG     HIGH        ;O
PIN    15   HSTB        REG     LOW         ;O
```

```
PIN     16    HRST          REG      LOW          ;O
PIN     17    VERT          REG      HIGH         ;O
PIN     18    VSTB          REG      LOW          ;O
PIN     19    VRST          REG      LOW          ;O
PIN     20    FRAME         REG      LOW          ;O
PIN     21    FRST          HIGH                  ;O
PIN     22    FWRITE        REG      LOW          ;O
PIN     23    IRQ           REG      LOW          ;O
PIN     24    VCC


EQUATIONS

HSTB = XCO                                  ; STROBE TO LATCH HORIZ SYNC
HSTB.CLKF = XCLK                            ; CLOCKED BY XCLK

HORIZ = VCC                                 ; HORIZ SYNC
HORIZ.CLKF = HSTB                           ; LATCHED BY HSTB
HORIZ.SETF = HRST                           ; RESET BY HRST

HRST = XC06                                 ; RESET FOR HORIZ SYNC
HRST.CLKF = XC02                            ; CLOCKED BY XC02 FOLLOWING XC06
HRST.RSTF = /HORIZ * HRST                   ; CLEARED BY HRST AND AFTER
                                            ;   HORIZ SYNC IS CLEARED

VSTB = YCO                                  ; STROBE TO LATCH VERTICAL SYNC
VSTB.CLKF = XCLK                            ; CLOCKED BY XCLK

VERT = VCC                                  ; VERTICAL SYNC
VERT.CLKF = VSTB                            ; LATCHED BY VSTB
VERT.SETF = VRST                            ; RESET BY VRST

VRST = VCC                                  ; RESET FOR VERTICAL SYNC
VRST.CLKF = YC02                            ; CLOCKED BY YC02 (2 LINES)
VRST.RSTF = /VERT * VRST                    ; CLEARED BY VRST AND AFTER
                                            ;   VERTICAL SYNC IS CLEARED

IRQ = INTEN                                 ; IRQ GENERATED ONLY WHEN ENABLED
IRQ.CLKF = VERT                             ; CLOCKED BY VERT
IRQ.RSTF = IMDTACK                          ; CLEARED BY IMDTACK

FRAME = VCC                                 ; FRAME MARKER WRITE STROBE
FRAME.CLKF = VERT                           ; CLOCKED BY VERT
FRAME.RSTF = FRST                           ; CLEARED BY FRST

FRST = /INTEN + FWRITE * XCLK               ; FRAME MARKER STROBE RESET WHEN
                                            ; INTERRUPTS NOT ENABLED AND AFTER
                                            ; WRITE PULSE

FWRITE = FRAME + PIX_ON0 + PIX_ON1          ; FIFO WRITE PULSE WHEN FRAME OR
                                            ;   ANY PIXELS ON
FWRITE.CLKF = /XCLK                         ; CLOCKED ON /XCLK (DELAY)
FWRITE.RSTF = FRST                          ; CLEARED IF INTERRUPTS NOT
                                            ;   ENABLED
```

```
SIMULATION

TRACE_ON XCLK XCO XC06 XC02 HSTB HORIZ HRST YCO YC02 VSTB VERT VRST
INTEN IMDTACK IRQ PIX_ON0 PIX_ON1 FRAME FWRITE FRST

SETF /PL OE /XCLK /YCO /YC02 /XCO /XC06 /XC02 INTEN /IMDTACK /PIX_ON0 /PIX_ON1

PRLDF VSTB /VERT VRST HSTB /HORIZ HRST IRQ FRAME FWRITE

; TEST HORIZ SYNC

SETF XCO
SETF XCLK
SETF /XCLK
CHECK HSTB
SETF /XCO
CHECK HORIZ
SETF XC06
SETF XCLK
SETF /XCLK
SETF XC02
SETF XCLK
SETF /XCLK
CHECK /HORIZ
SETF /XC06 /XC02
SETF XCLK
SETF /XCLK

; TEST VERTICAL SYNC AND IRQ

SETF INTEN
SETF XCLK
SETF /XCLK

SETF YCO
SETF XCLK
CHECK VSTB
SETF /YCO
CHECK VERT
CHECK IRQ FRAME
SETF /XCLK
CHECK FWRITE
SETF XCLK
SETF /XCLK
CHECK /FRAME
SETF YC02
SETF XCLK
SETF /XCLK
CHECK /VERT
SETF XCLK
SETF /XCLK
SETF /YC02
SETF XCLK
SETF /XCLK
SETF IMDTACK
```

```
CHECK /IRQ
SETF XCLK
SETF /XCLK
SETF /IMDTACK

; TEST INTEN ON VERTICAL SYNC

SETF /INTEN

SETF YCO
SETF XCLK
CHECK VSTB
CHECK VERT
CHECK /IRQ /FRAME
SETF /YCO
SETF XCLK
SETF /XCLK

; TEST PIX_ON0 WRITES

SETF INTEN

SETF XCLK
SETF PIX_ON0
CHECK /FWRITE
SETF /XCLK
CHECK FWRITE
SETF /PIX_ON0
SETF XCLK
CHECK /FWRITE
SETF /XCLK

SETF /INTEN

SETF XCLK
SETF PIX_ON0
SETF /XCLK
CHECK /FWRITE
SETF /PIX_ON0

; TEST PIX_ON1 WRITES

SETF INTEN

SETF XCLK
SETF PIX_ON1
CHECK /FWRITE
SETF /XCLK
CHECK FWRITE
SETF /PIX_ON1
SETF XCLK
CHECK /FWRITE
SETF /XCLK

SETF /INTEN
```

```
SETF XCLK
SETF PIX_ON1
SETF /XCLK
CHECK /FWRITE
SETF /PIX_ON1

TRACE_OFF
```

Appendix C

Calibration

All sensor and actuator signals are sampled by the Analog-to-Digital (A/D) and Digital-to-Analog (D/A) circuitry before being passed to the controller. The analog signals from the sensors and to the actuators are adjusted such that full-scale readings on the A/D's and D/A's correspond with the maximum outputs and inputs of the sensors and actuators, respectively. The controller, therefore, must provide at least a gain and an offset for each sensor and actuator signal to transform them into the proper units.

Nonlinearities in the sensors or actuators, however, can make a single gain and offset for each device inadequate. Calibration with a polynomial correction function can often reduce the effects of the nonlinearities. Three areas where polynomial fits made a significant improvement in the Multi-Manipulator Free-Flying Space Robot experiment are the camera-lens-distortion correction, the RVDT joint-angle nonlinearity compensation, and the motor-torque-curve compensation. Accordingly, this appendix is divided into three sections to show the improvement of a polynomial fit in each case.

C.1 Camera-Lens-Distortion Correction

The camera lens used in the global vision system is a 6mm wide-angle lens. There are, therefore, some barrel-distortion effects around the edges. Since the lens is circularly symmetric, this distortion is a function of the radius. To correct for this distortion, the calibration procedure performs a polynomial fit as a function of both the x and y coordinates. That is:

$$\begin{aligned} x_c &= f_x(x_m, y_m) \\ y_c &= f_y(x_m, y_m) \end{aligned} \tag{C.1}$$

where x_m and y_m represent the measured x and y coordinates of a point, x_c and y_c are the corrected values, and f_x and f_y are the polynomial correction functions. This correction is equivalent to performing the correction as a function of the radius, since the radius is itself a function of x and y:

$$\mathbf{r}_m = \sqrt{x_m^2 + y_m^2} \tag{C.2}$$

Moreover, performing this calibration also reduces errors caused by camera tilt and rotation.

A large plate containing a grid of 8×16 evenly-space LED's is used for calibration[1]. The LED's are spaced at 6 inches and a complete grid of 12×16 LED's is needed to cover the field of view of a camera. The calibration plate can be shifted to provide the full set of data, and the overlapping rows are used to accurately align the two calibration plate positions. The calibration routines find the coefficients of polynomials that minimize the errors between measured LED locations and the actual known locations of the LED's on the calibration plate.

The functions of a third-order polynomial fit, for example, have the form:

$$\begin{aligned} x_c &= c_{x_0}x_m^3 + c_{x_1}x_m^2 + c_{x_2}x_m + c_{x_3}y_m^3 + c_{x_4}y_m^2 + c_{x_5}y_m \\ &\quad + c_{x_6}x_m^2y_m + c_{x_7}x_my_m^2 + c_{x_8}x_my_m + c_{x_9} \end{aligned} \tag{C.3}$$

$$\begin{aligned} y_c &= c_{y_0}x_m^3 + c_{y_1}x_m^2 + c_{y_2}x_m + c_{y_3}y_m^3 + c_{y_4}y_m^2 + c_{y_5}y_m \\ &\quad + c_{y_6}x_m^2y_m + c_{y_7}x_my_m^2 + c_{y_8}x_my_m + c_{y_9} \end{aligned} \tag{C.4}$$

Given the polynomial order n, the number of coefficients is $M = (n+1)(n+2)/2$, thus the third-order fit has ten coefficients for each coordinate.

The coefficients are determined by solving a least-squares problem. Each row i of the regressor matrix $\mathbf{A}$ is formed using a measured data pair (x_{m_i}, y_{m_i}) as:

$$\begin{bmatrix} x_{m_i}^3 & x_{m_i}^2 & x_{m_i} & y_{m_i}^3 & y_{m_i}^2 & y_{m_i} & x_{m_i}^2y_{m_i} & x_{m_i}y_{m_i}^2 & x_{m_i}y_{m_i} & 1 \end{bmatrix} \tag{C.5}$$

Denoting the actual location of the ith LED as (x_{a_i}, y_{a_i}), the least-squares problem for a collection of N measured data sets is then formed as:

$$\underbrace{\begin{bmatrix} x_{a_1} & y_{a_1} \\ \vdots & \vdots \\ x_{a_N} & y_{a_N} \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} x_{m_1}^3 & x_{m_1}^2 & x_{m_1} & y_{m_1}^3 & y_{m_1}^2 & y_{m_1} & x_{m_1}^2y_{m_1} & x_{m_1}y_{m_1}^2 & x_{m_1}y_{m_1} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m_N}^3 & x_{m_N}^2 & x_{m_N} & y_{m_N}^3 & y_{m_N}^2 & y_{m_N} & x_{m_N}^2y_{m_N} & x_{m_N}y_{m_N}^2 & x_{m_N}y_{m_N} & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} c_{x_0} & c_{y_0} \\ \vdots & \vdots \\ c_{x_M} & c_{y_M} \end{bmatrix}}_{\mathbf{C}}$$

[1]The calibration plate is built by Kurt Zimmerman.

Solving the least-squares problem for $\mathbf{C}$ gives both sets of coefficients to use for lens-distortion correction. This can be performed easily with a single command in MATLAB: $\mathbf{C} = \mathbf{A}\backslash\mathbf{X}$.

Figure C.1 shows the effects of first-order, second-order, and third-order polynomial fits. The left column shows the actual LED locations as 'o's and the corrected locations as '+'s for each order fit. The right plots show, qualitatively, the errors in each fit in a three-dimensional view[2]. The error plots for the first- and second-order fits show clearly the circularly-symmetric effect of the lens distortion. The plots for the third-order fit show the dramatic improvement. Using even higher order fits does not significantly improve the fit, and has the disadvantage of having too many coefficients, slowing the computation.

The MATLAB code for performing the polynomial fits follow.

[2]The upper left data point in each error plot is a dummy value to preserve the scaling throughout the three plots.

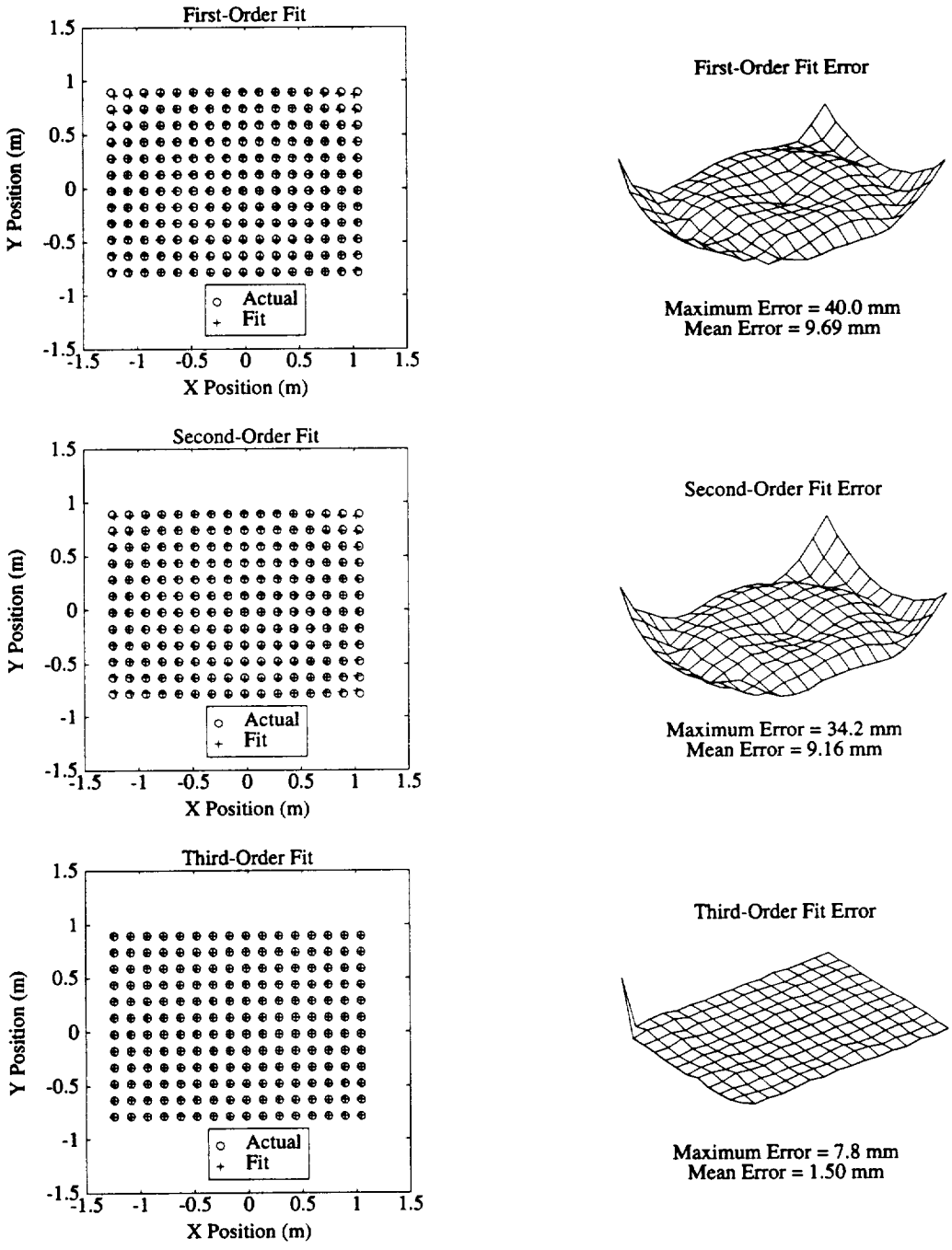


Figure C.1: Lens-Distortion Calibration

These are the lens-distortion correction results for three different polynomial fits. The left plots show the actual LED locations as the 'o's and the corrected measured locations as '+'s. It can be seen that there is significant improvement with the third-order polynomial. The right plots show the errors in a three-dimensional view. The upper-right corner of each error plot contains a dummy data point, used as a reference to preserve the scaling throughout all three plots.

C.1.1 Polynomial Fit to Two Variables

```
function p = polyfit2(x,y,n)
% POLYFIT2    POLYFIT2(x,y,n) finds the coefficients of
%             two-variable polynomials formed from the data in
%             matrix x of degree n that fits the data in matrix y in
%             a least-squares sense.  Each column of x represents a
%             variable, and a polynomial is fit to each column of y.
%


% Vincent Chen 1-4-90
% Revised 7-31-90 for only 2 variables w/ cross terms
% Copyright (c) 1990

% the regression problem is formulated in matrix format as:
%
% y = A*P    or
%
%        3   2       3   2       2       2
% y = [x1  x1  x1  x2  x2  x2  x1 x2 x1x2  x1x2 1] [p13
%                                                   p12
%                                                   p11
%                                                   p23
%                                                   p22
%                                                   p21
%                                                    .
%                                                    .
%                                                    .
%                                                   p0 ]
%
% where the matrix P contains the coefficients to be found.  Each column
% of P contains the coefficients for a fit to the corresponding column
% of y.  NOTE that there is only one constant term coefficient, p0.
% For a 7th order polynomial of a single variable, matrix A would be:
%
% A = [x.^7 x.^6 x.^5 x.^4 x.^3 x.^2 x ones(x)];
%
% See also polyval2

xs = size(x);
ys = size(y);
if ( xs(1) ~= ys(1) )
    error('X and Y must have the same number of rows')
end

if ( (xs(2) ~= 2) | (ys(2) ~= 2) )
    error('X and Y must have exactly 2 columns')
end

% A must have same number of rows as x and each column must be
% repeated n times
Ancol = 0;
for i = 0:n
    Ancol = Ancol + i + 1;
```

```
end

if ( xs(1) < Ancol )
    error('Not enough data to perform the fit, decrease n')
end

A = ones(xs(1), Ancol);

% Construct Vandermonde matrix
for i = 1:2
    for j=1:n
        A(:,n*(i-1)+j) = x(:,i).^(n+1-j);
    end
end


Aindex = 2*n + 1;
for i = 1:n-1
    for j = 2*n-(i-1):2*n
        A(:,Aindex) = A(:,i+1).*A(:,j);
        Aindex = Aindex + 1;
    end
end

p = (A\y);
```

C.1.2 Evaluation of Polynomial of Two Variables

```
function y = polyval2(c,x)
% POLYVAL2 Polynomial evaluation.
%       If C is a matrix whose column elements are the coefficients of a
%       polynomial, then POLYVAL2(C,X) is the value of the
% polynomials evaluated at X.  If X is a vector,
% the polynomial is evaluated at all points in X.
% If X is a matrix, then each column of X represents a variable
% in a two-variable polynomial, and each column of C contains
% the coefficients of the variables in succession.  For a two
% variable, 3rd order polynomial, a column of C looks like:
%
%               c = [c13
%                    c12
%                    c11
%                    c23
%                    c22
%                    c21
%                    c021
%                    c012
%                    c011
%                    c0]
%
%    and if X is [x1  x2], then
%
%                           3           2
```

```
%               y = c13 * x1   + c12 * x1   + c11 * x1
%
%                           3             2
%                 + c23 * x2   + c22 * x2   + c21 * x2
%
%                             2                         2
%                 + c021 * x1 * x2 + c012 * x1 * x2
%
%                 + c011 * x2 * x2
%
%                 + c0

% Polynomial evaluation c(x) using Horner's method

% Vincent Chen 1-5-90
% Revised
% Copyright 1990

% See also polyfit2

[m,n] = size(x);
[mc, nc] = size(c);

order = 0;
ocount = 1;
while ocount < mc,
    order = order + 1;
    ocount = ocount + order + 1;
end


if ( ocount ~= mc )
    error('column size of C does not match any order of polynomials')
end

if (m+n) == 2
    % Make it scream for scalar X.  Polynomial evaluation can be
    % implemented as a recursive digital filter.
    y = zeros(1,nc);
    for i = 1:nc
        yy = filter(1,[1 -x],c(:,i));
        y(1,i) = yy(nc);
    end
    return
end

% Do general case where X is an array
y = zeros(m,nc);
for j=1:n
    yy=zeros(m,nc);
    for i=1:order
        yy = (x(:,j) * ones(1,nc)) .* yy...
           + ones(m,1) * c(((j-1)*order)+i,:);
    end
    y = y + (x(:,j) * ones(1,nc)) .* yy;
```

```
end

cindex=2*order+1;

for i = order-1:-1:1
    for j = order-i:-1:1
        y = y + ((x(: ,1) * ones(1, nc)).^ i).* ((x(:,2) * ones(1, nc)).^ j).*...
        (ones(m, 1) * c(cindex,:));
        cindex = cindex + 1;
    end
end

y = y + ones(m,1) * c(mc,:);
```

C.2 Joint-Angle Calibration

The RVDT's used to measure joint angles also contain some nonlinearity. A simple third-order polynomial as a function of the a single variable—the measured joint-angle—is used for each joint RVDT.

A calibration jig, containing holes in know locations, is bolted to the robot base. The end effector of each manipulator is place in each of the holes for RVDT readings. These are matched with the joint angles derived from exact inverse-kinematics equations.

Figure C.2 shows the errors of using a linear fit vs. a third-order fit for each joint. In all cases, there is noticeable improvement.

C.3 Torque-Curve Calibration

The Aeroflex brushless DC motors used to actuate the space robot manipulators deliver smooth torques with very low friction. The motors are, however, limited-angle torquers, and the delivered torques for a given input current drop as the motors rotates away from zero angle. This effect is symmetric around the zero angle of each motor, and can be minimized by introducing a scaling function that is dependent on the motor angle. A scaling function essentially boosts the motor current to achieve the requested output torque for all motor angles.

The torque curve, i.e., scaling function, can be modelled as a fourth-order polynomial, restricted to having a single inflection point at the central motor angle. Since the controller has direct access to the joint-angles rather than motor angles, the calibration curves are derived as functions of the joint-angles.

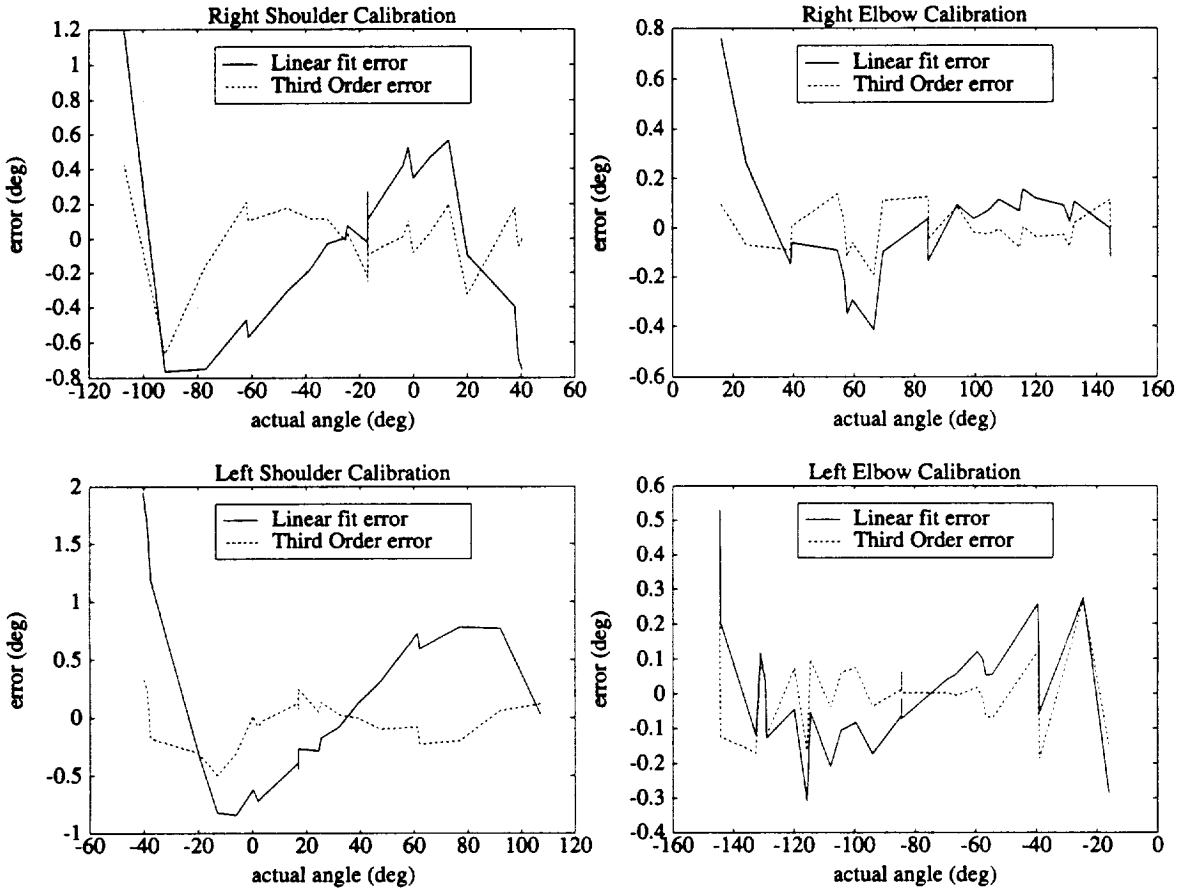


Figure C.2: **Joint-Angle Calibration Errors**

These plots show the calibration errors for a linear fit vs. a third-order fit for each joint. They clearly show the improvement with the third-order fits.

They have the form:

$$scale_i = c_{4i}\left(q_i - q_{\text{offset}i}\right)^4 + c_{2i}\left(q_i - q_{\text{offset}i}\right)^2 + 1 \tag{C.6}$$

where $q_{\text{offset}i}$ is the angle of joint i that corresponds to the zero angle of motor i, and c_{4i} and c_{2i} are the coefficients to be derived from least-squares fits[3]. The requested torques from the control laws are multiplied by these scale factors before sending them to the motor drivers.

Figures C.3 and C.4 show typical calibration curves. The data was taken by measuring the requested torques necessary to cancel the effects of a known applied torques at different joint angles. The 'o's

[3]Strictly speaking, c_{2i} should be zero to force a single inflection point in the torque curve, but its presence can improve the fit.

indicate the values of the applied torques, the '+'s indicate the torques that the control law requests, and the 'x's indicate the requested torques after fourth-order fit. For a perfect fit, the 'o's and the 'x's should overlap. The plots show that the fit is very reasonable.

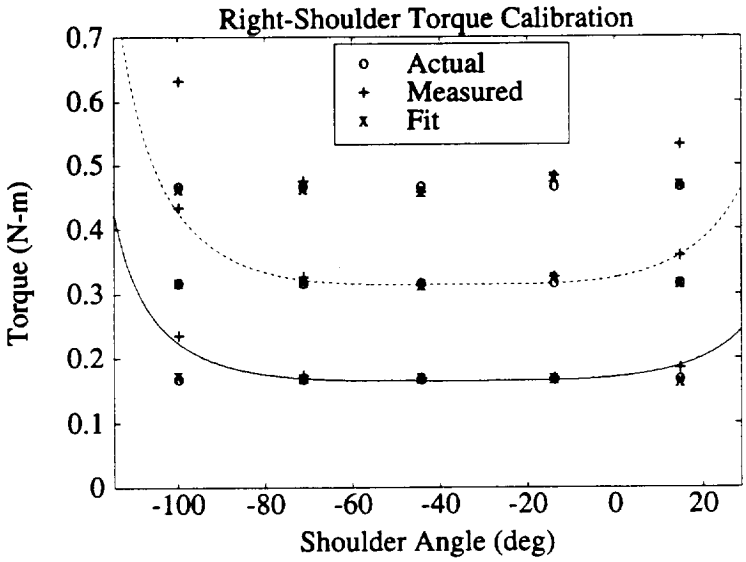


Figure C.3: **Right Shoulder Motor Calibration Curves**

The lines in Figures C.3 and C.4 plot the torque curve as a continuous function of the joint angle for the three values of applied torques. Ideally, these torque curves should intersect the '+' marks, and the figures show that the torque curves the torque curves do fit the measured values, indicating again that the fourth-order calibration polynomial is a good model for the behavior of the manipulator motors.

C.3.1 Torque Calibration File

```
% Name:
%    torquecal.m - motor torque calibration
%
% Description:
%    The first section lists the measured data and the rest performs
%    the calibration.
%
!rm *.met

if arm == 0 % Right Shoulder
 % Data
    t0=-[-.06 -.015 .01 .03 .05];
```

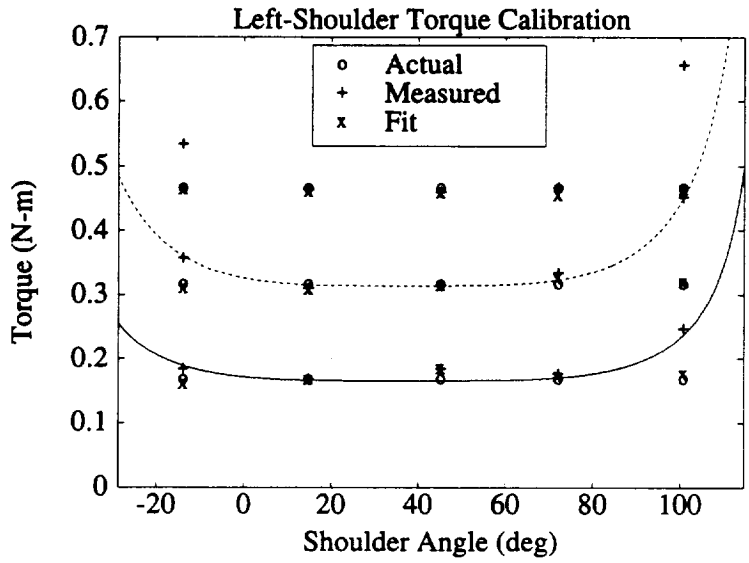


Figure C.4: **Left Shoulder Motor Calibration Curves**

```
    t1=-[.22 .19 .21 .23 .27];
    t2=-[.46 .375 .385 .42 .48];
    t3=-[.70 .555 .56 .61 .69];
 % The angles at which the data were taken
    ang0=[-1.75 -1.25 -.78 -.25 .25];
 % Gain for the motor
    sc = -0.8272;
 % Joint angle corresponding to motor's zero angle
    q0 = -.65;
 % Plotting axes and title text
    axis([-2*180/pi .5*180/pi 0 .7]);
    tl='Right';
else       % Left Shoulder
 % Data
    t0=-[-.05 -.04 .0 .04 .1];
    t1=-[-.28 -.25 -.23 -.18 -.21];
    t2=-[-.5 -.43 -.395 -.38 -.47];
    t3=-[-.725 -.625 -.58 -.55 -.73];
 % The angles at which the data were taken
    ang0=[-.25 .25 .78 1.25 1.75];
 % Gain for the motor
    sc = .7872;
 % Joint angle corresponding to motor's zero angle
    q0 = .65;
 % Plotting axes and title text
    axis([-.5*180/pi 2*180/pi 0 .7]);
    tl='Left';
end
```

```
% Set up data stream, subtracting the effects of the spring forces.
t=[t1-t0 t2-t0 t3-t0]'*sc;
tu=[t1 t2 t3]'*sc;

% Set up the ideal data set.
ideal0=[0 0 0 0 0];
ideal1=.165*ones(1,5);
ideal2=.314*ones(1,5);
ideal3=.464*ones(1,5);

ideal=[ideal1 ideal2 ideal3]';

ang=[ang0 ang0 ang0]';

% Perform the fit using either:
%        c4 x^4 + c2 x^2 + c0
% or
%        c4 x^4 + c0
%
if fourth == 0
    c=polyfit((ang-ones(ang)*q0).^2, ideal./t, 2);
    c1=[c(1) 0 c(2) 0 c(3)];
    cor=t.*polyval(c1, (ang-ones(ang)*q0));
    coru=tu.*polyval(c1, (ang-ones(ang)*q0));
else
    c=polyfit((ang-ones(ang)*q0).^4, ideal./t, 1);
    c1=[c(1) 0 0 0 c(2)];
    cor=t.*polyval(c1, (ang-ones(ang)*q0));
    coru=tu.*polyval(c1, (ang-ones(ang)*q0));
end

% Setting up for plots
names=[ 'Actual  '
        'Measured'
        'Fit     '];
ltypes=[ 'o'
         '+'
         'x'];

plot(ang*180/pi, t, '+b'); hold on; grid;
plot(ang*180/pi, cor, 'xr');
plot(ang*180/pi, ideal, 'ow');
err=cor-ideal;
maxerr=max(err);

a=[-4:.01:4];
s=polyval(c1,a);

plot([a;a;a]'*180/pi, ([.165;.314;.464]*(ones(a)./polyval(c1, a-ones(a)*q0)))');
hold off
title([tl '-Shoulder Torque Calibration'])
xlabel('Shoulder Angle (deg)'),ylabel('Torque (N-m)')
c=legndbig(names,0,0,1000,1,ltypes);
pause
```

Appendix D

State Transition Diagrams

This appendix contains the state-transition graphs for the Multi-Manipulator Free-Flying Space Robot experiment. Implementing the transition graphs allows the robot to complete complicated tasks autonomously. Although not explicitly indicated, all control is performed utilizing the adaptive *task*-space control structure.

The graphs depict the states as ovals, and the state transitions as arrows. A stimulus-and-transition-routine pair is labeled at the base of each arrow; the pair is separated by a "/" character. If a transition routine returns more than one value, causing a branch in the transition, the return values are labeled near the arrow heads.

The state-transition graphs are separated into six (6) main functions: "initialization", "payload capture", "rendezvous with object", "deliver object", "object motion", and "robot motion". Whenever the robot is enabled, it executes the initialization sequence, checking the health of the system. The robot carries out a payload capture in response to user request; if the payload object is not in range, the robot first performs a rendezvous with the object. If the user requests the movement of the payload, the robot executes the delivery of the object. It must also execute the "object motion" transitions to check the final orientation of the payload; if the orientation is not achievable with the current manipulator grasp, a swap is performed. Finally, if the user requests a base movement, the robot executes the "robot motion" transitions.

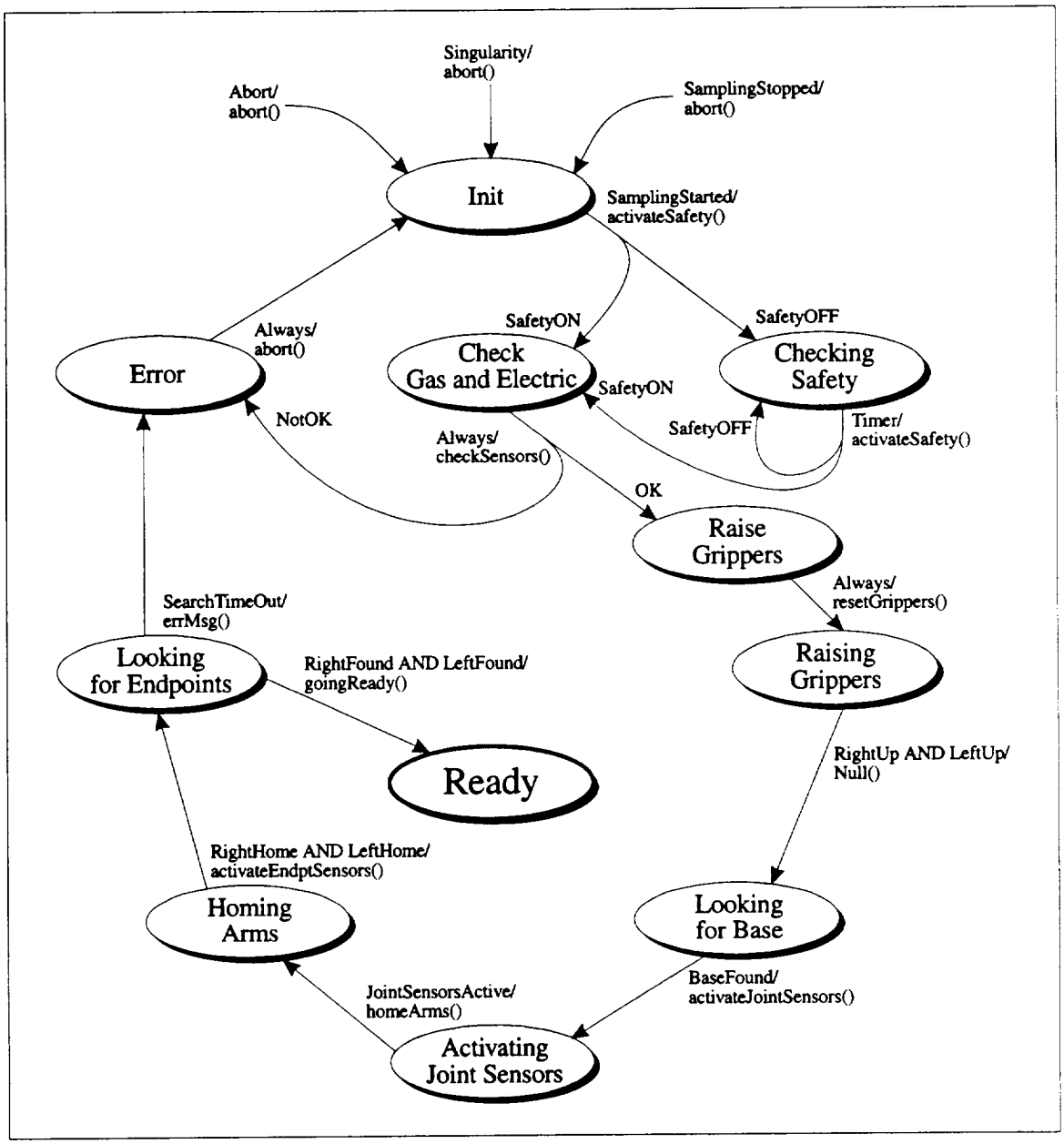


Figure D.1: **Initialization Transition Graph**

The robot system starts in the "Init" state and checks the safety switch, gas pressure, and electrical voltage when sampling is started. When everything checks out, the grippers are raised and the system makes sure that the robot base is being tracked. The motors and joint controller are then enabled to bring the arms into a known "home" location. The local vision system must find the endpoints before entering the "Ready" state. If an error occurs, the robot enters the "Error" state, which transitions back into the "Init" state, and the process repeats. Additionally, three global stimuli—Abort, Singularity, and SamplingStopped—brings the system to "Init" from any other state.

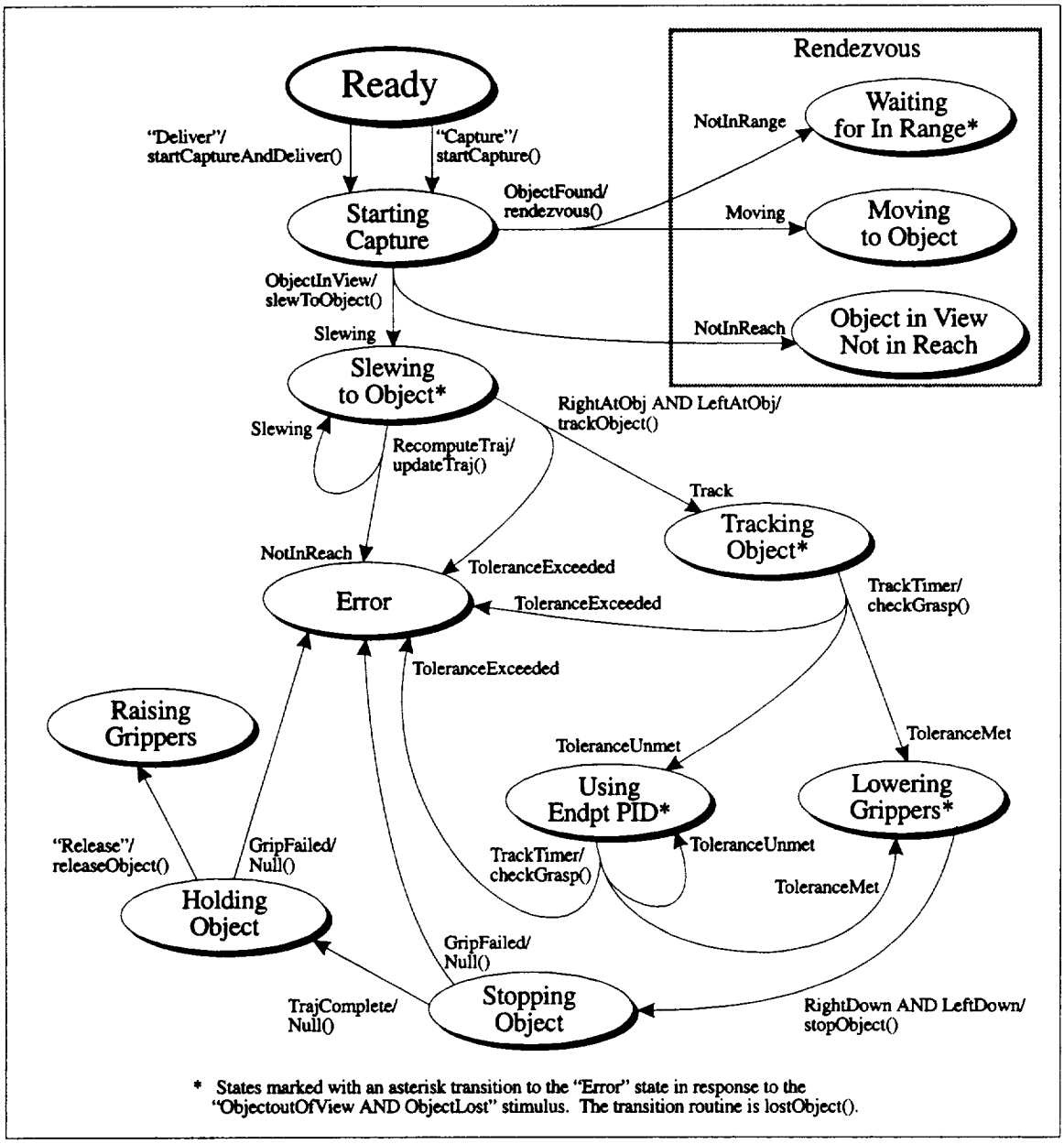


Figure D.2: Payload-Capture Transition Graph

The robot awaits the "Capture" or "Deliver" command in the "Ready" state, either of which initiates the payload-capture transition routines. If the object is not in reach, the robot branches to the "Rendezvous" set of transitions. Otherwise, the robot slews the arms to the object—tracking the grip ports using PID control until the tolerance is met—before lowering the grippers. The object is stopped smoothly, ending in the "Holding Object" state. If the command was "Deliver", the robot proceeds with the "Object Delivery" transitions. A "Release" command forces the transition from "Holding Object" to "Raising Grippers", which is part of the "Initialization" transition graph.

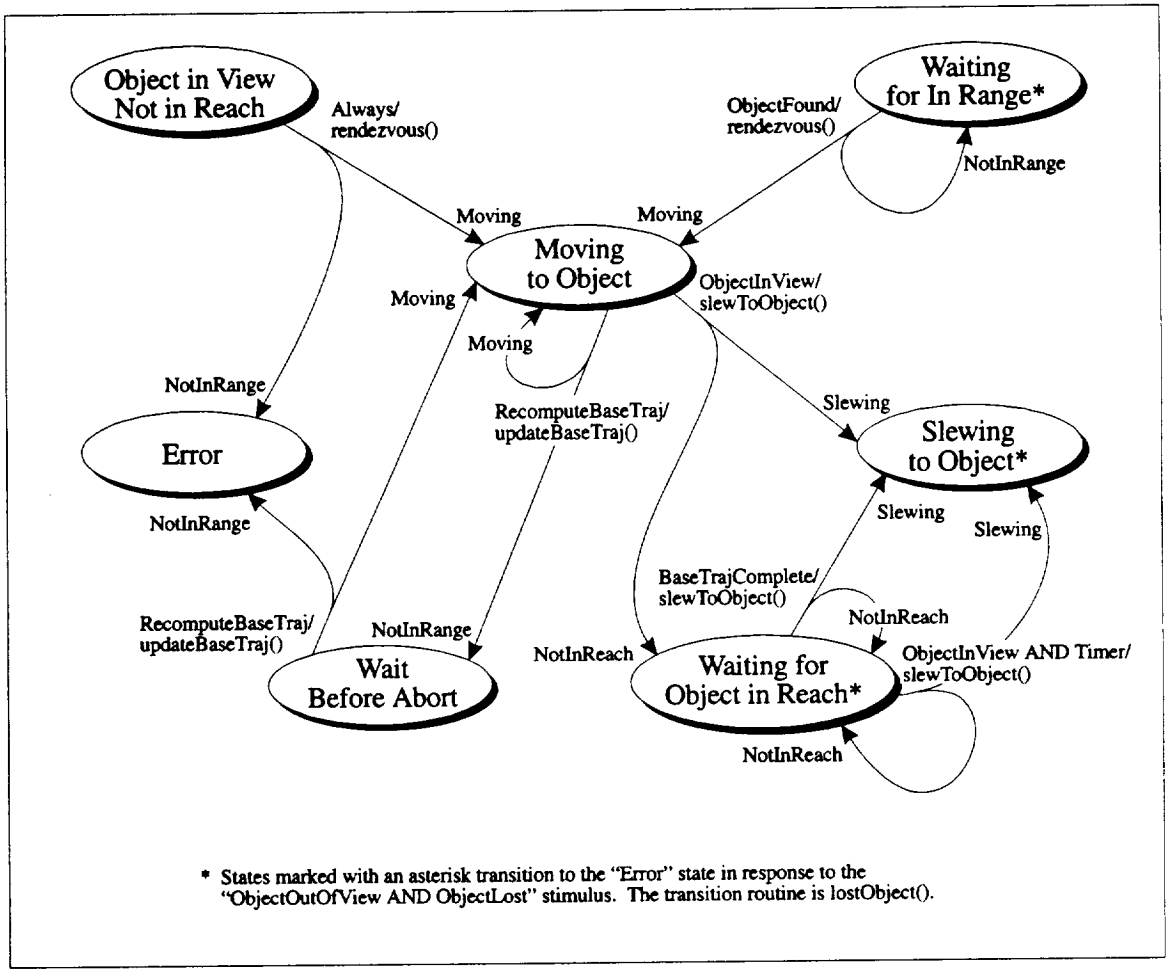


Figure D.3: **Object-Rendezvous Transition Graph**

The robot must rendezvous with the object if it is out of reach. The robot plots an intercept trajectory, and moves toward the object. The robot continues to update the base trajectory until the object comes into view of the local vision system. If at any time, the robot decides that it cannot reach the object, the robot aborts into the "Error" state. When the object comes into view, the robot waits for the the object to come into reach and slews the manipulators to the object and enters the "Payload-Capture" transition graph. An "Error" occurs if the object is lost by the vision systems.

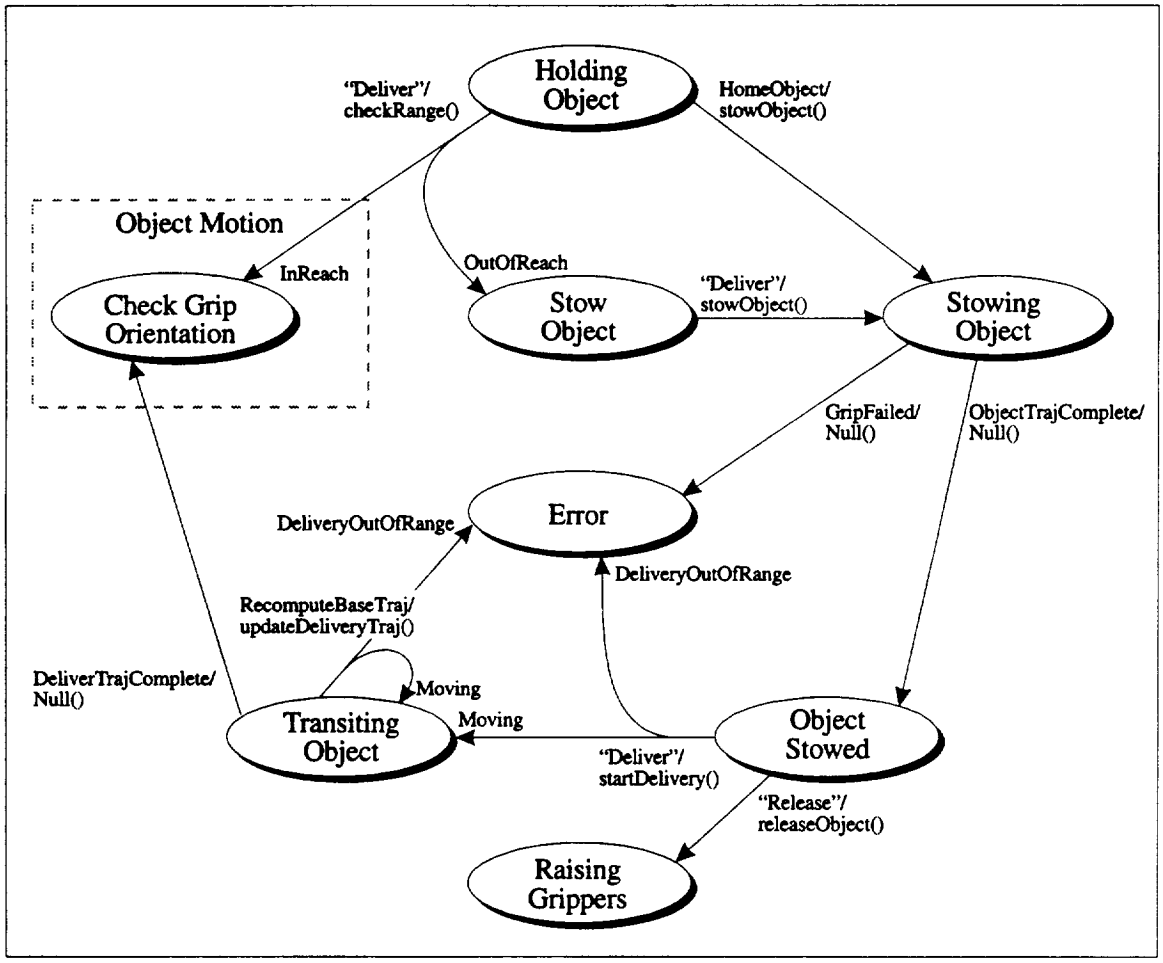


Figure D.4: **Object-Delivery Transition Graph**

If the robot is to deliver the payload and the destination is in the reach of the manipulators, the robot goes directly to the "Object-Motion" set of transitions. Otherwise, it must stow the object before plotting a base trajectory toward the destination. The robot continues to update the base trajectory until it reaches the destination. If at any time, the robot determines that the destination is out of range, it aborts with an "Error" condition. After reaching the destination of the robot base, the robot proceeds with the "Object-Motion" transitions.

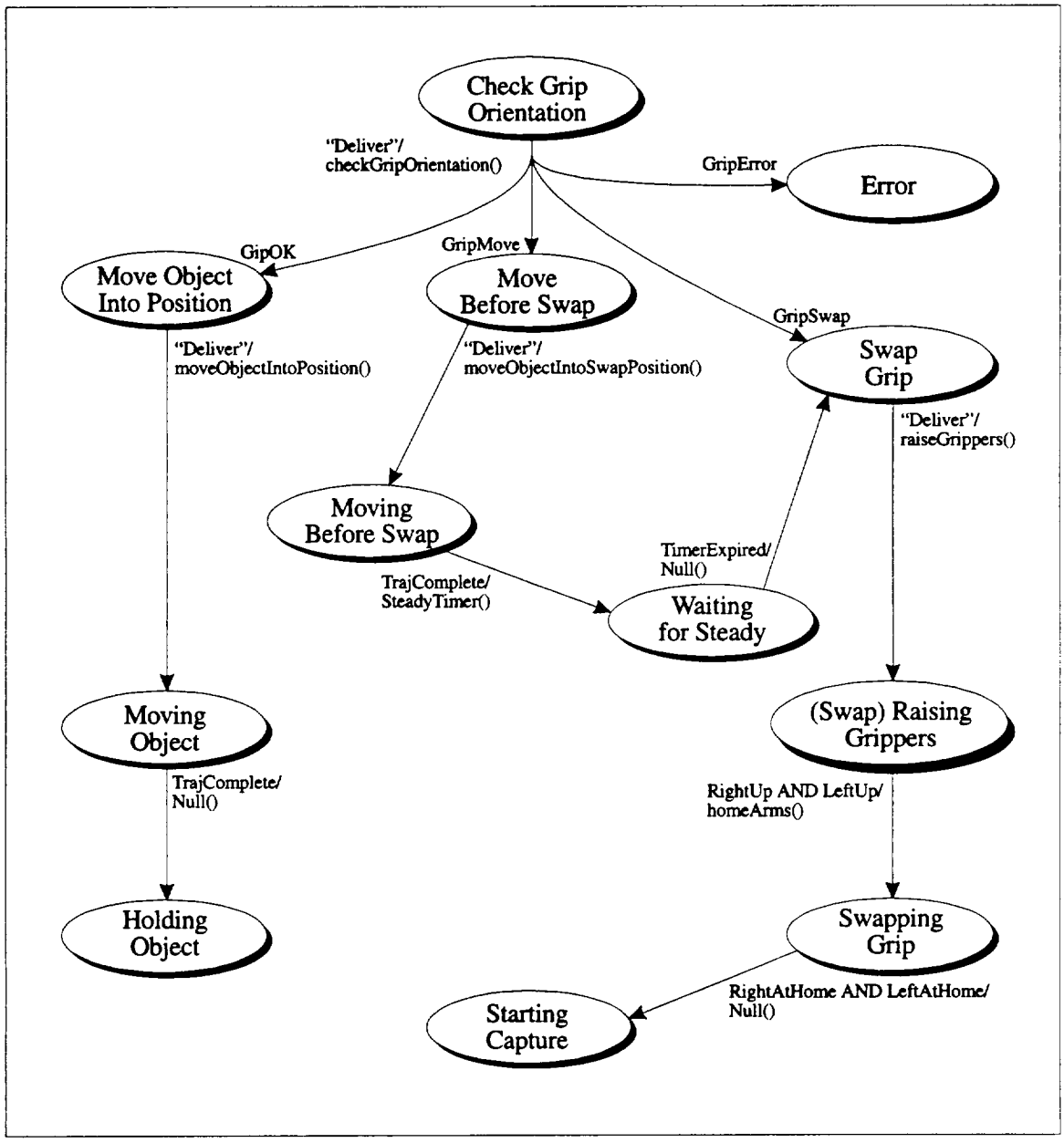


Figure D.5: **Object-Motion Transition Graph**

The first step in moving the payload object into its final position is to check the grip orientation. If it is achievable, the robot directly moves the payload into position and returns to the "Holding Object" state. Otherwise, the robot either swaps the grip directly, or first moves the payload into a favorable position before performing the swap. The swap is performed by releasing the object, homing the arms, and recapturing the payload by looping back to the "Starting Capture" state. Because the "Deliver" command is still in effect, the complete capture-and-delivery procedure is repeated.

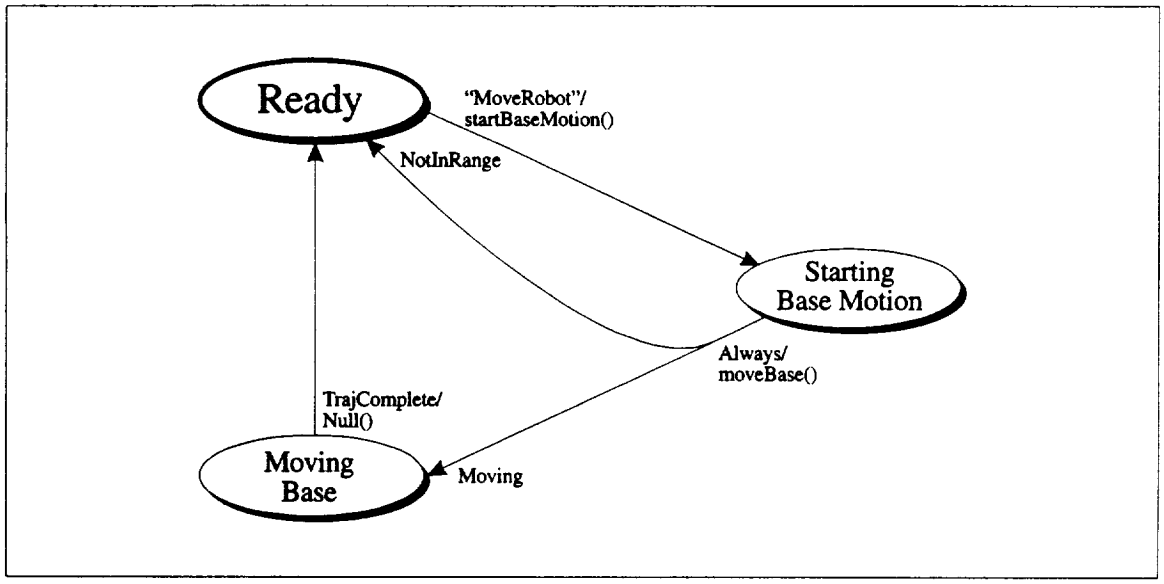


Figure D.6: **Robot-Motion Transition Graph**

This is the simplest transition of all. If the destination is reachable, the trajectory is computed, and the robot base is moved. Otherwise, the command is ignored. The robot ends up in the "Ready" state.

Appendix E

Setup Files

This appendix provides a sampling of the setup files used by the Multi-Manipulator Free-Flying Space Robot during initialization. These files contain, for instance, the nominal physical parameters for the robot, the adaptive update gains, and the controller gains for each control mode. Many more data files are required to provide the full capabilities of the space robot.

E.1 Physical Parameters

```
#  Name:
#       properties.dat
#
#  Description:
#       physical properties of the robot
#
#  Written by:  Vincent Chen                                    November 1990
#_____________________________________________________________________________

signalSet          Masses
    BaseMass              kg       62.265
    RAUpperLinkMass       kg       1.9231
    RALowerLinkMass       kg       0.3382
    LAUpperLinkMass       kg       1.9231
    LALowerLinkMass       kg       0.3382
    PayloadMass           kg       1.01
    LargePayloadMass      kg       8.87

signalSet          Inertias
    BaseInertia           kg-m^2   3.29218
    RAUpperLinkInertia    kg-m^2   0.02379
    RALowerLinkInertia    kg-m^2   0.00416
    LAUpperLinkInertia    kg-m^2   0.02379
    LALowerLinkInertia    kg-m^2   0.00416
```

```
    PayloadInertia      kg-m^2  0.007
    LargePayloadInertia kg-m^2  0.108

signalSet         Lengths
    L_C1x               m       0.184095
    L_C1y               m      -0.184095
    L_C2x               m       0.184095
    L_C2y               m       0.184095
    L_11                m       0.3048
    L_12                m       0.2959
    L_21                m       0.3048
    L_22                m       0.2959
    L_C0x               m       0
    L_C0y               m       0
    L_11x               m       0.0594
    L_11y               m      -0.002
    L_12x               m       0.1058
    L_12y               m       0
    L_21x               m       0.0594
    L_21y               m       0.002
    L_22x               m       0.1058
    L_22y               m       0
    L_OBJx              m       0
    L_OBJy              m       0
    L_GRIPrx            m       0.111
    L_GRIPry            m       0
    L_GRIPlx            m      -0.111
    L_GRIPly            m       0

signalSet        SpringConstants
    RS_SpringK          N-m/rad 0
    RE_SpringK          N-m/rad 0
    LS_SpringK          N-m/rad 0
    LE_SpringK          N-m/rad 0
    RS_SpringOffset     rad    -.78
    RE_SpringOffset     rad    1.57
    LS_SpringOffset     rad     .78
    LE_SpringOffset     rad   -1.57
```

E.2 Adaptive Control Gains

```
#  Name:
#       BTAdaptSigset.dat
#
#  Description:
#       Adaptive Controller parameter vector and gains.
#
#  Written by:  Vincent Chen                          26 October 1990
#_____________________________________________________________________

signalSet Parameters    28
    Theta_0
    Theta_1
```

```
    Theta_2
    Theta_3
    Theta_4
    Theta_5
    Theta_6
    Theta_7
    Theta_8
    Theta_9
    Theta_10
    Theta_11
    Theta_12
    Theta_13
    Theta_14
    Theta_15
    Theta_16
    Theta_17
    Theta_18
    Theta_19
    Theta_20
    Theta_21
    Theta_22
    Theta_23
    Theta_24
    Theta_25
    Theta_26
    Theta_27

signalSet ParametersMax 28
    ThetaMax_0  100
    ThetaMax_1  2
    ThetaMax_2  2
    ThetaMax_3  1
    ThetaMax_4  .1
    ThetaMax_5  1
    ThetaMax_6  .1
    ThetaMax_7  1
    ThetaMax_8  .1
    ThetaMax_9  1
    ThetaMax_10 .1
    ThetaMax_11 10
    ThetaMax_12 1
    ThetaMax_13 .1
    ThetaMax_14 1
    ThetaMax_15 .1
    ThetaMax_16 20
    ThetaMax_17 .1
    ThetaMax_18 .1
    ThetaMax_19 1
    ThetaMax_20 .05
    ThetaMax_21 .05
    ThetaMax_22 .05
    ThetaMax_23 .05
    ThetaMax_24 .05
    ThetaMax_25 .05
    ThetaMax_26 .05
```

```
    ThetaMax_27 .05

signalSet ParametersMin 28
    ThetaMin_0  0
    ThetaMin_1  -2
    ThetaMin_2  -2
    ThetaMin_3  0
    ThetaMin_4  -.1
    ThetaMin_5  0
    ThetaMin_6  -.1
    ThetaMin_7  0
    ThetaMin_8  -.1
    ThetaMin_9  0
    ThetaMin_10 -.1
    ThetaMin_11 0
    ThetaMin_12 0
    ThetaMin_13 0
    ThetaMin_14 0
    ThetaMin_15 0
    ThetaMin_16 0
    ThetaMin_17 -.1
    ThetaMin_18 -.1
    ThetaMin_19 0
    ThetaMin_20 -.05
    ThetaMin_21 -.05
    ThetaMin_22 -.05
    ThetaMin_23 -.05
    ThetaMin_24 -.05
    ThetaMin_25 -.05
    ThetaMin_26 -.05
    ThetaMin_27 -.05

signalSet ParameterAdaptiveGains   28
    Gamma_0     0
    Gamma_1     0
    Gamma_2     0
    Gamma_3     0
    Gamma_4     0
    Gamma_5     0
    Gamma_6     0
    Gamma_7     0
    Gamma_8     0
    Gamma_9     0
    Gamma_10    0
    Gamma_11    0
    Gamma_12    0
    Gamma_13    0
    Gamma_14    0
    Gamma_15    0
    Gamma_16    250
    Gamma_17    .01
    Gamma_18    .01
    Gamma_19    .06
    Gamma_20    0
    Gamma_21    0
```

```
    Gamma_22    0
    Gamma_23    0
    Gamma_24    0
    Gamma_25    0
    Gamma_26    0
    Gamma_27    0

signalSet SpringUpdateGains    8
    RSK         0
    RSO         0
    REK         0
    REO         0
    LSK         0
    LSO         0
    LEK         0
    LEO         0

signalSet BTNumParmsUpdate     1
    nParmsUpdate       28

# trailing was 1
adaptiveDataSet BTRobotParms   28        Sample
    nUpdate:    BTNumParmsUpdate
    trailing:   1
    params:     Parameters
    maxParams:  ParametersMax
    minParams:  ParametersMin
    gains:      ParameterAdaptiveGains
    errors:     JinvFilteredErrors
    regressor:  Yy
    transpose:  1
```

E.3 Base-Relative Object-Control Gains

```
#   Name:
#       objectControlBR.dat
#
#   Description:
#       Base-relative object control signals and gains.
#
#   Written by: Vincent Chen                        May 1991
#_____________________________________________________________

signalSet       DesObjectPosBR          2
    desObjectPosXBR     meters
    desObjectPosYBR     meters

signalSet       DesObjectOrientBR       1
    desObjectOrientBR   radians

signalSet       DesObjectVelBR          2
    desObjectVelXBR     m/sec
    desObjectVelYBR     m/sec
```

```
signalSet       DesObjectAngVelBR       1
    desObjectAngVelBR   rad/sec

signalSet       DesObjectAccBR          2
    desObjectAccXBR     m/sec^2
    desObjectAccYBR     m/sec^2

signalSet       DesObjectAngAccBR       1
    desObjectAngAccBR   rad/sec^2

signalSet       ObjectPosGainsBR        2
    objectPosGainXBR    55
    objectPosGainYBR    55

# Note velocity gain selected for critical damping: Kv = 2 * sqrt(Kp)
signalSet       ObjectVelGainsBR        2
    objectVelGainXBR    15
    objectVelGainYBR    15


# Note: These gains for the object are unstable if you use the 2Hz estimator
#       and vision-based velocity feedback.  A 5Hz estimator works fine.
signalSet       ObjectOrientGainsBR     1
    objectOrientGainBR  .55

# Note velocity gain selected for critical damping: Kv = 2 * sqrt(Kp)
signalSet       ObjectAngVelGainsBR     1
    objectAngVelGainBR  .15

signalSet       ObjectPosFeedbackBR     2
    objectPosFeedbackXBR
    objectPosFeedbackYBR

signalSet       ObjectOrientFeedbackBR  1
    objectOrientFeedbackBR

signalSet       ObjectPosControlBR      2
    objectPosControlXBR         m/sec^2
    objectPosControlYBR         m/sec^2

signalSet       ObjectOrientControlBR   1
    objectOrientControlBR       1/sec^2

signalSet       ObjectDestBRPos         2
    objectDestBRPosX    meters  0
    objectDestBRPosY    meters  0

signalSet       ObjectWorkSpaceCenter   2
    objBaseX    meters          0.48
    objBaseY    meters          0.0
```

E.4 Endpoint-Control Gains

```
#   Name:
#       endptControl.dat
#
#   Description:
#       Endpoint control gains and signals.
#
#_____________________________________________________________________

signalSet       RAdesEndptPos           2
    RAdesEndptXPos      meters  0
    RAdesEndptYPos      meters  0.2

signalSet       RAdesEndptVel           2
    RAdesEndptXVel      m/sec
    RAdesEndptYVel      m/sec

signalSet       RAdesEndptAcc           2
    RAdesEndptXAcc      m/sec^2
    RAdesEndptYAcc      m/sec^2

signalSet       LAdesEndptPos           2
    LAdesEndptXPos      meters  0
    LAdesEndptYPos      meters  -0.2

signalSet       LAdesEndptVel           2
    LAdesEndptXVel      m/sec
    LAdesEndptYVel      m/sec

signalSet       LAdesEndptAcc           2
    LAdesEndptXAcc      m/sec^2
    LAdesEndptYAcc      m/sec^2

signalSet       RAendptPosGains         2
    RAendptPosXGain             20
    RAendptPosYGain             20

signalSet       RAendptVelGains         2
    RAendptVelXGain             6
    RAendptVelYGain             6

signalSet       RAendptIntGains         2
    RAendptIntXGain             40
    RAendptIntYGain             40

signalSet       LAendptPosGains         2
    LAendptPosXGain             20
    LAendptPosYGain             20

signalSet       LAendptVelGains         2
    LAendptVelXGain             6
    LAendptVelYGain             6

signalSet       LAendptIntGains         2
```

```
    LAendptIntXGain             40
    LAendptIntYGain             40

signalSet       RAendptFeedback         2
    RAendptXfeedback
    RAendptYfeedback

signalSet       LAendptFeedback         2
    LAendptXfeedback
    LAendptYfeedback

signalSet       RAendptControl  2
    RAendptXcontrol     m/sec^2
    RAendptYcontrol     m/sec^2

signalSet       LAendptControl  2
    LAendptXcontrol     m/sec^2
    LAendptYcontrol     m/sec^2
```

E.5 Joint-Control Gains

```
#   Name:
#       jointPDcontrol.dat
#
#   Description:
#
#   Written by: Vincent Chen                        November 1990
#   Revised:    Vincent Chen                        May 1991
#       Modified for BTController.
#
#_____________________________________________________________________

signalSet  RAJointAngleGains    2
    RAShlderPosGain             8
    RAElbowPosGain              2

signalSet  LAJointAngleGains    2
    LAShlderPosGain             8
    LAElbowPosGain              2

signalSet  RAJointRateGains     2
    RAShlderVelGain             0.7
    RAElbowVelGain              0.5

signalSet  LAJointRateGains     2
    LAShlderVelGain             0.7
    LAElbowVelGain              0.5

signalSet  RAdesJointAngles     2
    RAdesRtShlderPos    rad     -1
    RAdesRtElbowPos     rad     1

signalSet  LAdesJointAngles     2
```

```
    LAdesRtShlderPos    rad     1
    LAdesRtElbowPos     rad     -1

signalSet  RAdesJointRates      2
    RAdesRtShlderVel    rad/sec 0
    RAdesRtElbowVel     rad/sec 0

signalSet  LAdesJointRates      2
    LAdesRtShlderVel    rad/sec 0
    LAdesRtElbowVel     rad/sec 0

signalSet       RAdesJointAccel         2
    RAdesShlderAcc      rad/sec^2
    RAdesElbowAcc       rad/sec^2

signalSet       LAdesJointAccel         2
    LAdesShlderAcc      rad/sec^2
    LAdesElbowAcc       rad/sec^2


PDdataSet       RAJointPDcontroller     2       Sample
    pos:        RAJointAngles
    vel:        RAJointRates
    output:     RAJointTorques
    Kp:         RAJointAngleGains
    Kv:         RAJointRateGains
    desPos:     RAdesJointAngles
    desVel:     RAdesJointRates
    orientation:        0

PDdataSet       LAJointPDcontroller     2       Sample
    pos:        LAJointAngles
    vel:        LAJointRates
    output:     LAJointTorques
    Kp:         LAJointAngleGains
    Kv:         LAJointRateGains
    desPos:     LAdesJointAngles
    desVel:     LAdesJointRates
    orientation:        0
```

E.6 Base-Control Gains

```
#   Name:
#       baseControl.dat
#
#   Revised:    Vincent Chen                                    May 1991
#       Modified for own use.
#-------------------------------------------------------------------------

signalSet       DesBasePos      2
    desBasePosX         meters
    desBasePosY         meters
```

```
signalSet       DesBaseOrient   1
    desBaseOrient       radians

signalSet       DesBaseVel      2
    desBaseVelX         m/sec
    desBaseVelY         m/sec

signalSet       DesBaseAngVel   1
    desBaseAngVel       rad/sec

signalSet       DesBaseAcc      2
    desBaseAccX         m/sec^2
    desBaseAccY         m/sec^2

signalSet       DesBaseAngAcc   1
    desBaseAngAcc       rad/sec^2

signalSet       BasePosGains    2
    basePosGainX        31
    basePosGainY        31

signalSet       BaseVelGains    2
    baseVelGainX        90
    baseVelGainY        90


signalSet       BaseOrientGains 1
    baseOrientGain      0.6

signalSet       BaseAngVelGains 1
    baseAngVelGain      2.0

signalSet       BasePosFeedback 2
    basePosFeedbackX
    basePosFeedbackY

signalSet       BaseOrientFeedback      1
    baseOrientFeedback

signalSet       BasePosControl  2
    basePosControlX     m/sec^2
    basePosControlY     m/sec^2

signalSet       BaseOrientControl       1
    baseOrientControl   1/sec^2
```

Bibliography

[1] Harold L. Alexander. *Experiments in Control of Satellite Manipulators.* PhD thesis, Stanford University, Department of Electrical Engineering, Stanford, CA 94305, December 1987.

[2] Chae H. An, Christopher G. Atkeson, and John M. Hollerbach. Estimation of Inertial Parameters of Rigid Body Links of Manipulators. In *Proceedings of 24th Conference on Decision and Control,* pages 990–995, Ft. Lauderdale, FL, December 1985. IEEE.

[3] Tom M. Apostol. *Calculus,* volume II. John Wiley & Sons, New York, NY, second edition, 1969.

[4] C. G. Atkeson, C. H. An, and J. M. Hollerbach. Estimation of Inertial Parameters of Manipulator Loads and Links. *International Journal of Robotics Research,* 5(3):101–18, Fall 1986.

[5] D. S. Bayard and J. T. Wen. New Class of Control Laws for Robotic Manipulators. II. Adaptive Case. *International Journal of Control,* 47(5):1387–406, May 1988.

[6] David S. Bayard and John T. Wen. Robust Control for Robotic Manipulators, Part II: Adaptive Case. EM 347-87-204, JPL Engineering Memorandum, 1987.

[7] Robert H. Cannon, Jr. and Thomas O. Binford. AFOSR Final Report of the Center For Automation and Manufacturing Science on Basic Research in Robotics. Final report, Stanford University Aerospace Robotics Laboratory, Stanford, CA 94305, October 1989.

[8] Craig R. Carignan. *Control Strategies for Manipulating Payloads in Weightlessness with a Free-Flying Robot.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, MIT, MA, September 1987.

[9] John J. Craig. *Adaptive Control of Mechanical Manipulators.* PhD thesis, Stanford University, Department of Electrical Engineering, Stanford, CA 94305, June 1986. To be published by Addison-Wesley.

[10] William C. Dickson. *Experiments in Cooperative Object Manipulation by Mobile Robot Teams.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, (September) 1992. To be Published.

[11] Gene F. Franklin, J. David Powell, and Michael L. Workman. *Digital Control of Dynamic Systems.* Addison-Wesley, Reading, MA, second edition, 1990.

[12] S. Hayati. Hybrid position/force control of multi-arm cooperating robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 82–89, San Francisco, CA, April 1986. IEEE, IEEE Computer Society.

[13] T. C. Hsia. Adaptive Control of Robot Manipulators—A Review. In *Proceedings of the International Conference on Robotics and Automation*, pages 183–189, San Francisco, CA, April 1986. IEEE, IEEE Computer Society.

[14] Yan-Ru Hu and A. A. Goldenberg. An Adaptive Approach to Motion and Force Control of Multiple Coordinated Robot Arms. In *Proceedings of the International Conference on Robotics and Automation*, pages 1091–1096, Scottsdale, AZ, April 1989. IEEE, IEEE Computer Society.

[15] Thomas R. Kane and David A. Levinson. *Dynamics: Theory and Application.* McGraw-Hill Series in Mechanical Engineering. McGraw-Hill, New York, NY, 1985.

[16] Oussama Khatib. Object Manipulation in a Multi-Effector Robot System. In *Proceedings IV of the International Symposium of Robotics Research*, Santa Cruz, CA, 1987.

[17] Pradeep K. Khosla and Takeo Kanade. Parameter Identification of Robot Dynamics. In *Proceedings of 24th Conference on Decision and Control*, Fort Lauderdale, FL, December 1985. IEEE.

[18] Dongmin Kim, Michael Walker, and Joseph Dionise. Adaptive Coordinated Motion Control of Two Manipulator Arms. In *Proceedings of the International Conference on Robotics and Automation*, pages 1084–1089, Scottsdale, AZ, April 1989. IEEE, IEEE Computer Society.

[19] Ross Koningstein. *Experiments in Cooperative-Arm Object Manipulation with a Two-Armed Free-Flying Robot.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, October 1990. Also published as SUDAAR 597.

[20] Ross Koningstein, Marc Ullman, and Robert H. Cannon, Jr. Computed Torque Control of a Free-Flying Cooperating-Arm Robot. In *Proceedings of the NASA Conference on Space Telerobotics,* Pasadena, CA, February 1989.

[21] Kader Laroussi, Hooshang Hemami, and Ralph Goddard. Coordination of Two Planar Robots in Lifting. *IEEE Journal of Robotics and Automation,* 4(1):77–85, February 1988.

[22] Weiping Li and Jean-Jacques E. Slotine. Parameter Estimation Strategies for Robotic Applications. In *A.S.M.E. Annual winter Meeting - Modeling and Control of Robotic Manipulators and Manufacturing Processes,* pages 213–218, Boston, MA, 1987.

[23] D. R. Meldrum, G. Rodriguez, and G. F. Franklin. An Order (N) Recursive Inversion of the Jacobian for an N-link Serial Manipulator. In *Proceedings of the International Conference on Robotics and Automation,* pages 1175–1180, Sacramento, CA, April 1991. IEEE, IEEE Computer Society.

[24] Diedra Meldrum. *Indirect Adaptive Control of a Rigid Multi-Link Serial Manipulator.* PhD thesis, Stanford University, Stanford, CA 94305, (December) 1992. To be published.

[25] Y. Nakamura, K. Nagai, and T. Yoshikawa. Mechanics of Coordinative Manipulation by Multiple Robotic Mechanisms. In *Proceedings of the International Conference on Robotics and Automation,* pages 991–998, Raleigh, NC, April 1987. IEEE, IEEE Computer Society.

[26] Romeo Ortega and Mark W. Spong. Adaptive Motion Control of Rigid Robots: a Tutorial. *Automatica,* 25(6):877–888, 1989.

[27] V. M. Popov. *Hyperstability of Control Systems.* Springer-Verlag, New York, 1973.

[28] Real-Time Innovations, Inc., 954 Aster, Sunnyvale, CA 94086. *ControlShell: Object-Oriented Framework for Real-Time System Software User's Manual,* 4.0a edition, June 1991.

[29] G. Rodriguez. Kalman Filtering, Smoothing, and Recursive Robot Arm Forward and Inverse Dynamics. *IEEE Journal of Robotics and Automation,* RA-3(6), December 1987.

[30] G. Rodriguez and R. E. Scheid, Jr. Recursive Inverse Kinematics for Robot Arms Via Kalman Filtering and Bryson-Frazier Smoothing. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, August 1987.

[31] Daniel Mark Rovner. *Experiments in Adaptive Control of a Very Flexible One-Link Manipulator.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, August 1987.

[32] S. Schneider. *Experiments in the Dynamic and Strategic Control of Cooperating Manipulators.* PhD thesis, Stanford University, Stanford, CA 94305, September 1989. Also published as SUDAAR 586.

[33] S. Schneider and R. H. Cannon, Jr. Object Impedance Control for Cooperative Manipulation: Theory and Experimental Results. *IEEE Journal of Robotics and Automation*, 8(3), June 1992. Paper number B90145.

[34] S. A. Schneider, M. A. Ullman, and V. W. Chen. ControlShell: A Real-Time Software Framework. In *Proceedings of the 1991 IEEE International Conference on Systems Engineering*, Dayton, OH, August 1991.

[35] H. Seraji. Configuration Control of Redundant Manipulators: Theory and Implementation. *IEEE Transactions on Robotics and Automation*, 5(4):472–90, August 1989.

[36] Michael D. Sidman. *Adaptive Control of a Flexible Structure.* PhD thesis, Stanford University, Department of Electrical Engineering, Stanford, CA 94305, June 1986. Also published as SUDAAR 556.

[37] Jean-Jacques E. Slotine and Weiping Li. On The Adaptive Control of Robot Manipulators. In *Proceedings of the ASME Winter Annual Meeting*, pages 51–56, Anaheim, CA, 1986.

[38] Jean-Jacques E. Slotine and Weiping Li. Adaptive Strategies in Constrained Manipulation. In *Proceedings of the International Conference on Robotics and Automation*, pages 595–603, Raleigh, NC, April 1987. IEEE, IEEE Computer Society.

[39] Jean-Jacques E. Slotine and Weiping Li. Theoretical Issues in Adaptive Manipulator Control. In *The Fifth Yale Workshop on Applications of Adaptive Systems Theory*, May 1987.

[40] M. Uchiyama and P. Dauchez. A Symmetric Hybrid Position/Force Control Scheme for the Coordination of a Two-Arm Robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 350–357, Philadelphia, PA, April 1988. IEEE, IEEE Computer Society.

[41] M. Uchiyama, N. Iwasawa, and K. Hakomori. Hybrid Position/Force Control for Coordination of a Two-Arm Robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 1242–1247, Raleigh, NC, April 1987. IEEE, IEEE Computer Society.

[42] Christopher R. Uhlik. *Experiments in High-Performance Nonlinear and Adaptive Control of a Two-Link, Flexible-Drive-Train Manipulator.* PhD thesis, Stanford University, Department of Electrical Engineering, Stanford, CA 94305, May 1990. Also published as SUDAAR 592.

[43] M. A. Ullman. *Experiments in Autonomous Navigation and Control of Multi-Manipulator Free-Flying Space Robots.* PhD thesis, Stanford University, Stanford, CA 94305, (July) 1992. To be published.

[44] Y. Umetani and K. Yoshida. Continuous Path Control of Space Manipulators Mounted on OMV. *ACTA Astronautica*, 15(12):981–986, 87.

[45] Yoji Umetani and Kazuya Yoshida. Experimental Study on Two-Dimensional Free-Flying Robot Satellite Model. In *Proceedings of the NASA Conference on Space Telerobotics*, Pasadena, CA, January 1989.

[46] R. L. Vasquez. *Experiments in Two-Cooperating-Arm Manipulation from a Platform with Unknown Motion.* PhD thesis, Stanford University, Department of Aeronautics and Astronautics, Stanford, CA 94305, January 1992. Also published as SUDAAR 617.

[47] J. T. Wen. A Unified Perspective on Robot Control: the Energy Lyapunov Function Approach. In *Proceedings of the 29th Conference on Decision and Control*, volume 3, pages 1968–1973, Honolulu, HI, December 1991. IEEE.

[48] John T. Wen and David S. Bayard. Robust Control for Robotic Manipulators, Part I: Non-adaptive Case. EM 347-87-203, JPL Engineering Memorandum, 1987.

[49] D. E. Whitney and J. L. Nevins. What is the remote centre compliance (RCC) and what can it do? *Robot Sensors*, 2:3–15, 1986.

[50] Roberto Zanutta. *Experiments in Adaptive Control of Cooperating Manipulators.* PhD thesis, Stanford University, Department of Mechanical Engineering, Stanford, CA 94305, January 1991.

[51] Y. F. Zheng and J. Y. S. Luh. Optimal Load Distribution for Two Robots Handling a Single Object. In *Proceedings of the International Conference on Robotics and Automation*, pages 344–349, Philadelphia, PA, April 1988. IEEE, IEEE Computer Society.